

UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET



Miloš Milaković

SEKVENCIranJE ANTIBIOTIKA -
ELEKTRONSKA LEKCIJA

master rad

Beograd, 2025.

Mentor:

dr Jovana KOVAČEVIĆ, vanredni profesor
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

dr Mirjana MALJKOVIĆ RUŽIČIĆ, docent
Univerzitet u Beogradu, Matematički fakultet

Nevena ĆIRIĆ, asistent
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: _____

Najvoljenijima

Naslov master rada: Sekvenciranje antibiotika - Elektronska lekcija

Rezime: Ovaj rad predstavlja elektronsku lekciju posvećenu sekvenciranju antibiotika, sa fokusom na interaktivno upoznavanje sa različitim algoritamskim pristupima. Lekcija obuhvata teorijsko objašnjenje i vizuelizaciju sledećih algoritama za sekvenciranje proteina: gruba sila, *Branch and Bound*, *Leaderboard*, spektralna konvolucija i *DeepNovo* sekvenciranje. Korisnicima je omogućeno da prate izvršavanje algoritama korak po korak, sa opcijama pauziranja i ponavljanja, čime se olakšava razumevanje kompleksnih procesa sekvenciranja. Ova interaktivna platforma može služiti kao edukativni alat za studente i predavače u oblasti bioinformatike.

Ključne reči: sekvenciranje, algoritmi, računarstvo, aminokiseline, maseni spektrometar, gruba sila, *Branch and Bound*, *Leaderboard*, *DeepNovo*, spektralna konvolucija

Sadržaj

1	Uvod	1
1.1	Cilj rada	1
2	Teorijske osnove	4
2.1	Centralna dogma molekularne biologije	4
2.2	Odstupanje od centralne dogme	6
2.3	Maseni spektrometar	8
2.4	Teorijski spektar peptida	9
3	Algoritmi za sekvenciranje	11
3.1	Pristup grubom silom (<i>Brute Force</i>)	11
3.2	<i>Branch and Bound</i>	13
3.3	Uporedna analiza prethodnih algoritama	16
3.3.1	Pristup grubom silom	16
3.3.2	Algoritam <i>Branch and Bound</i>	17
3.3.3	Zaključak	17
3.4	Nedostak prethodnih algoritama	18
3.5	<i>Leaderboard</i> algoritam	18
3.5.1	Prednosti algoritma	18
3.5.2	Primene	19
3.6	Spektralna konvolucija	21
3.7	<i>DeepNovo</i>	24
3.7.1	Računska složenost	24
3.7.2	Tehnike zasnovane na <i>De Novo</i> sekvenciranju	25
3.7.3	Opšti pregled	25
3.7.4	Arhitektura neuronske mreže	26
3.7.5	Rezultati i evaluacija	29

4	Elektronska platforma	31
4.1	Pokretanje aplikacije	31
4.1.1	<i>Docker Compose</i> konfiguracija	31
4.1.2	Direktno pokretanje komponenti	33
4.1.3	<i>Google Cloud Run</i> implementacija	35
4.2	Funkcionalnosti platforme	38
4.2.1	Početna stranica	38
4.2.2	Uvodna stranica	39
4.2.3	Pristup grubom silom	41
4.2.4	<i>Branch and Bound</i>	44
4.2.5	<i>Leaderboard</i>	45
4.2.6	Spektralna konvolucija	48
4.2.7	<i>DeepNovo</i>	50
4.2.8	Poređenje algoritama	51
4.2.9	Nepostojeća stranica	53
5	Zaključak	55
	Bibliografija	56

Glava 1

Uvod

Sekvenciranje bioloških molekula predstavlja proces određivanja preciznog redosleda gradivnih jedinica od kojih su sastavljeni. Tako, na primer, govorimo o sekvenciranju DNK, kojim se utvrđuje niz nukleotida u genomu organizma. Međutim, osim genoma, moguće je sekvencirati i proteine, pri čemu se određuje redosled aminokiselina koji čini njihovu primarnu strukturu. Takav postupak naziva se sekvenciranje proteina.

U prirodi postoji 20 standardnih aminokiselina (Slika 1.1) koje se kombinuju u različite sekvence kako bi formirale proteine. U kontekstu antibiotika, specifičan redosled aminokiselina je od presudnog značaja jer direktno utiče na trodimenzionalnu strukturu molekula, biološku aktivnost i mehanizam dejstva.

Mnogi savremeni antibiotici predstavljaju peptide, što znači da su zapravo kratki lanci aminokiselina, tj. mali proteini. Antibiotici predstavljaju hemijska jedinjenja koja uništavaju mikroorganizme ili inhibiraju njihov rast, čime imaju ključnu ulogu u borbi protiv infekcija. Zbog njihove biološke važnosti i sve izraženijeg problema antimikrobne rezistencije, sekvenciranje antibiotika predstavlja značajnu oblast istraživanja u savremenoj bioinformatici i farmaceutskoj industriji. Zbog toga, ovaj rad će se fokusirati upravo na proces sekvenciranja antibiotika.

1.1 Cilj rada

U ovom radu implementirana je interaktivna elektronska platforma namenjena prikazu algoritamskih pristupa za sekvenciranje antibiotika, sa fokusom na njihovu vizuelizaciju u realnom vremenu. Platforma omogućava da se istraže principi i rad

Aminokiselina	Skraćenica	Masa (Da)
Glycine	G	57
Alanine	A	71
Serine	S	87
Proline	P	97
Valine	V	99
Threonine	T	101
Cysteine	C	103
Isoleucine	I	113
Leucine	L	113
Asparagine	N	114
Aspartic Acid	D	115
Lysine	K	128
Glutamine	Q	128
Glutamic Acid	E	129
Methionine	M	131
Histidine	H	137
Phenylalanine	F	147
Arginine	R	156
Tyrosine	Y	163
Tryptophan	W	186

Slika 1.1: Esencijalne aminokiseline sa svojim masama izraženim u daltonima (Da)

savremenih algoritama kroz interaktivne simulacije, čime se olakšava razumevanje kompleksnih bioinformatičkih koncepata.

Glavni ciljevi ovog rada su:

- Pružiti pregled modernih algoritama za sekvenciranje antibiotika
- Razviti interaktivnu edukativnu platformu za vizuelizaciju algoritama za se-

kvenciranje proteina

Rad je posebno koristan studentima bioinformatike i molekularne biologije kao alat za bolje razumevanje algoritama za rekonstrukciju peptidnih sekvenci.

Glava 2

Teorijske osnove

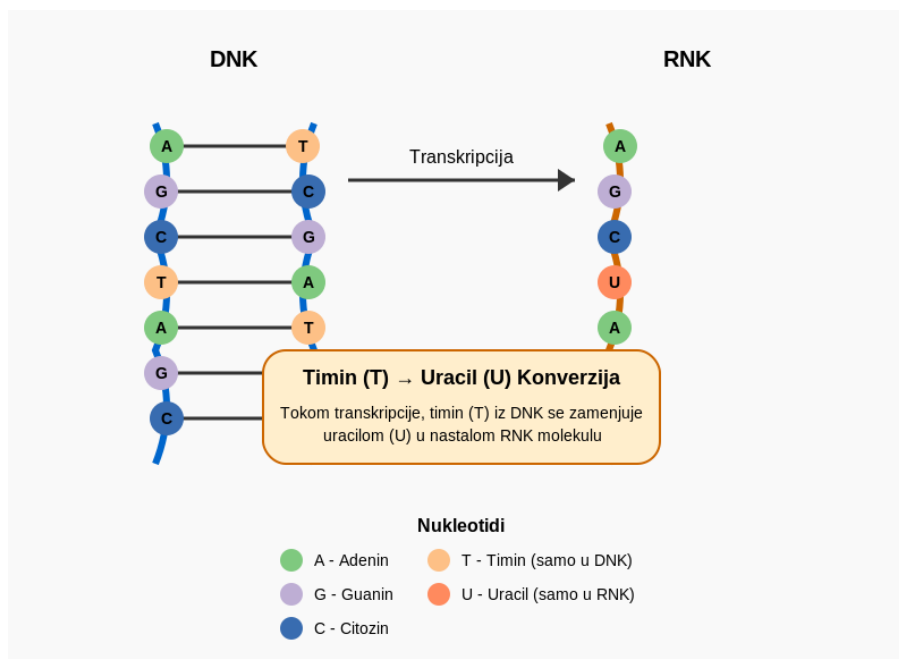
2.1 Centralna dogma molekularne biologije

Proces sekvenciranja antibiotika je fundamentalan u razumevanju kako su ovi molekuli proizvedeni od strane bakterija i kako se oni mogu sintetisati ili modifikovati za primene u medicini. Antibiotici su često peptidi, pri čemu veliki broj pripada neribozomalnim peptidima (*non-ribosomal peptides* - *NRPs*), koji ne prate standardna pravila za sintezu proteina čime se otežava njihovo sekvenciranje [12, 14].

DNK sadrži recept za kreiranje proteina. Naime, DNK se sastoji od gena koji mogu biti uključeni i tada će se na osnovu njih kreirati proteini ili isključeni kada se oni neće koristiti za kreiranje proteina. Ovaj proces uključivanja i isključivanja gena naziva se genska ekspresija i zavisi od toga da li je potrebno da se određeni protein kreira ili ne (na primer, fotosinteza kod biljaka koja se obavlja samo preko dana).

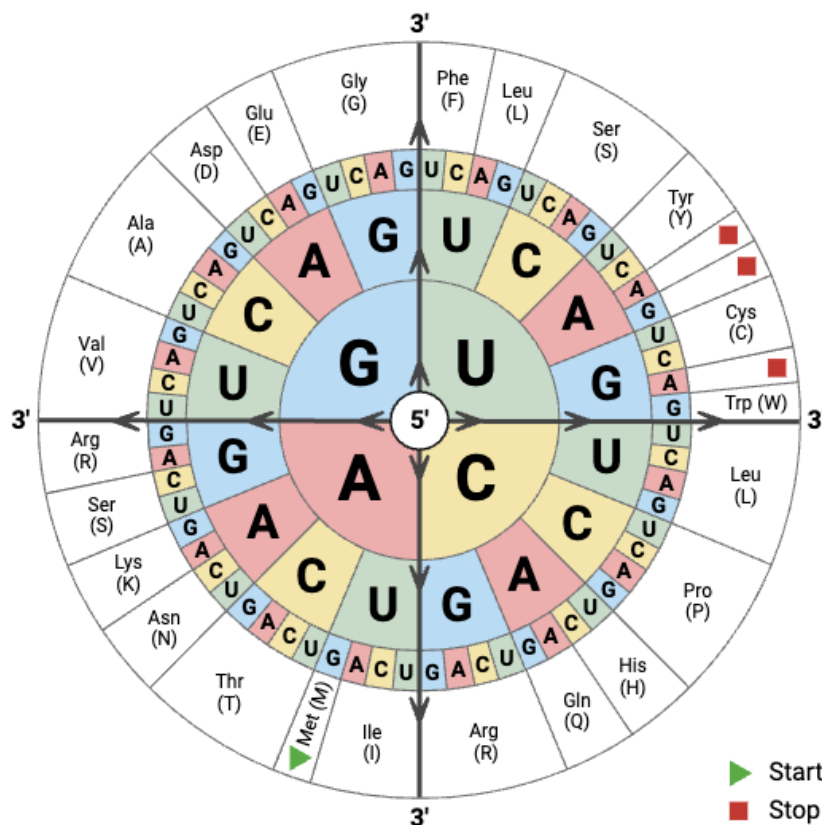
Najveći broj proteina nastaje kroz proces koji se zove centralna dogma molekularne biologije, koja podrazumeva da se DNK prvo prepisuje u RNK (Slika 2.1), a zatim se RNK prevodi u protein. Proces prevođenja DNK u RNK naziva se transkripcija i on podrazumeva da enzim RNK polimeraza očitava jedan lanac DNK i na osnovu njega sintetiše jedan, komplementaran RNK lanac, pri čemu se nukleotid timin (T) zamenjuje uracilom (U). Na Slici 2.1 se može se primetiti da se DNK sastoji od 2 lanca koja su komplementarna. Enzim RNK polimeraza se zakači na DNK lanac na početku gena, zatim se kreće duž gena čime razdvaja dva lanca DNK i tako stvara prostor za prepisivanje DNK čime se dobija novi molekul RNK.

Prilikom prevođenja RNK u protein potrebno je na osnovu nukleotida odrediti koja je aminokiselina u pitanju. Za ovaj proces je zadužena organela ribozom, koja omogućava očitavanje informacija koju nosi RNK prema unapred utvrđenom pravilu



Slika 2.1: Transkripcija DNK u RNK

koje se naziva genetski kod. Budući da je potrebno uniformno tumačiti nukleotidnu sekvencu, koristi se niz od tri uzastopna nukleotida, poznat kao kodon. Upotrebom kodona dužine tri nukleotida dobija se ukupno 64 različite sekvence, koje se mapiraju na 20 različitih aminokiselina. Da je korišćen niz od samo dva nukleotida, bilo bi moguće formirati svega 16 kombinacija, što ne bi bilo dovoljno da se pokriju sve aminokiseline neophodne za sintezu proteina. Na Slici 2.2 može se videti kako se kodoni prevode u odgovarajuće aminokiseline. Postoje start i stop kodoni koji određuju početak odnosno kraj sekvence koja se prevodi u protein.

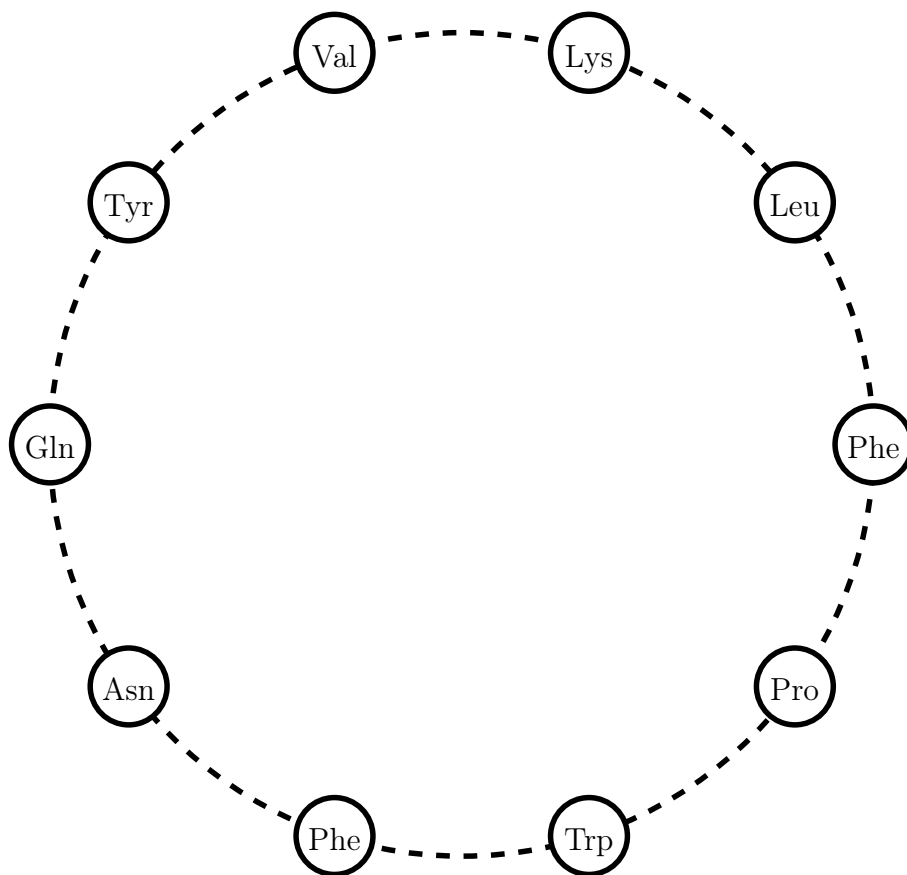


Slika 2.2: RNK kodonski točak prikazuje kako se sekvence od tri nukleotida (kodoni) prevode u aminokiseline. Svaki kodon se čita od centra ka spolja, a zeleni trougao označava start kodon (AUG) koji kodira metionin, dok crveni kvadrati označavaju stop kodone (UAA, UAG, UGA) koji određuju kraj sekvence koja se prevodi u protein. Preuzeto sa [15].

2.2 Odstupanje od centralne dogme

Tiroidin B1 je cikličan peptid dužine 10 (Slika 2.3), što znači da su prva i poslednja aminokiselina povezane i da samim tim postoji 10 njegovih različitih linearnih reprezentacija (Tabela 2.1). Prateći centralnu dogmu i činjenicu da se 1 kodon prevodi u 1 aminokiselinu, bilo je za očekivati da je moguće pronaći 10 kodona odnosno 30 nukleotida u genomu bakterije *Bacillus brevis* od koje nastaje ovaj antibiotik. Da bi se ovaj postupak obavio, potrebno je proveriti više hiljada 30-grama koji mogu da počnu bilo gde u genomu, što može delovati kao dugotrajan zadatak, ali uz moderne

računare je ostvariv u prihvatljivom vremenu. Analiziranjem genoma utvrđeno je da ne postoji 30-gram koji se kodira u neki od 10 različitih reprenzatacija traženog antibiotika.



Slika 2.3: Struktura tirocidina B1, cikličnog peptida sastavljenog od 10 aminokiselina.

Na ovaj način je pokazano da Tirocidin B1 ne prati centralnu dogmu molekularne biologije i da ovaj peptid nastaje kroz drugačiji proces. Dalja istraživanja pokazala su da postoje peptidi koji nastaju uz pomoć posebnih enzima koji su zaduženi za njihovo sintentisanje pod nazivom *NRP* sintetaze. Ovi enzimi se sastoje od kompleksnih molekulskih jedinica, takozvanih modula, gde svaki modul odgovara jednoj aminokiselini koja učestvuje u sastavu proteina. U slučaju Tirocidina B1, enzim sadrži 10 modula i svaki od modula kodira 1 aminokiselinu čime je određena struktura antibiotika.

#	Linearna sekvenca
1	Lys – Leu – Phe – Pro – Trp – Phe – Asn – Gln – Tyr – Val
2	Leu – Phe – Pro – Trp – Phe – Asn – Gln – Tyr – Val – Lys
3	Phe – Pro – Trp – Phe – Asn – Gln – Tyr – Val – Lys – Leu
4	Pro – Trp – Phe – Asn – Gln – Tyr – Val – Lys – Leu – Phe
5	Trp – Phe – Asn – Gln – Tyr – Val – Lys – Leu – Phe – Pro
6	Phe – Asn – Gln – Tyr – Val – Lys – Leu – Phe – Pro – Trp
7	Asn – Gln – Tyr – Val – Lys – Leu – Phe – Pro – Trp – Phe
8	Gln – Tyr – Val – Lys – Leu – Phe – Pro – Trp – Phe – Asn
9	Tyr – Val – Lys – Leu – Phe – Pro – Trp – Phe – Asn – Gln
10	Val – Lys – Leu – Phe – Pro – Trp – Phe – Asn – Gln – Tyr

Tabela 2.1: Deset različitih linearnih reprezentacija tirocidina B1

Samim tim, pošto struktura proteina nije određena na osnovu genoma bakterije, metode za sekvencioniranje DNK ovde nisu od pomoći i potrebno je sekvencirati direktno sam peptid.

2.3 Maseni spektrometar

Maseni spektrometar je mašina za merenje masa molekula, uključujući mase peptida i proteina. Alat funkcioniše na principu analize više uzoraka istog peptida, tako što date uzorke deli na sve moguće potpeptide, nakon čega se određuju mase potpeptida. U realnosti uzorak se pretvara u naelektrisanе jone da bi na njima mogli da utiču električno i magnetno polje. Potom se joni dele na osnovu odnosa mase i naelektrisanja, a njihove vrednosti se zatim precizno mere [11].

Masa se meri u daltonima (Da), pri čemu je 1 Da približno jednak masi protona/neutrona. Samim tim masa molekula je jednaka sumi masa protona/neutrona koji čine taj molekul. Mase aminokiselina su poznate i prikazane su na Slici 1.1. Može se primetiti da neke aminokiseline imaju istu masu, tako da 20 različitih aminokiselina ima 18 različitih masa.

Samim tim na osnovu poznatih masa aminokiselina možemo da odredimo masu celog peptida. Na primer, masa tirocidina se može izračunati na sledeći način:

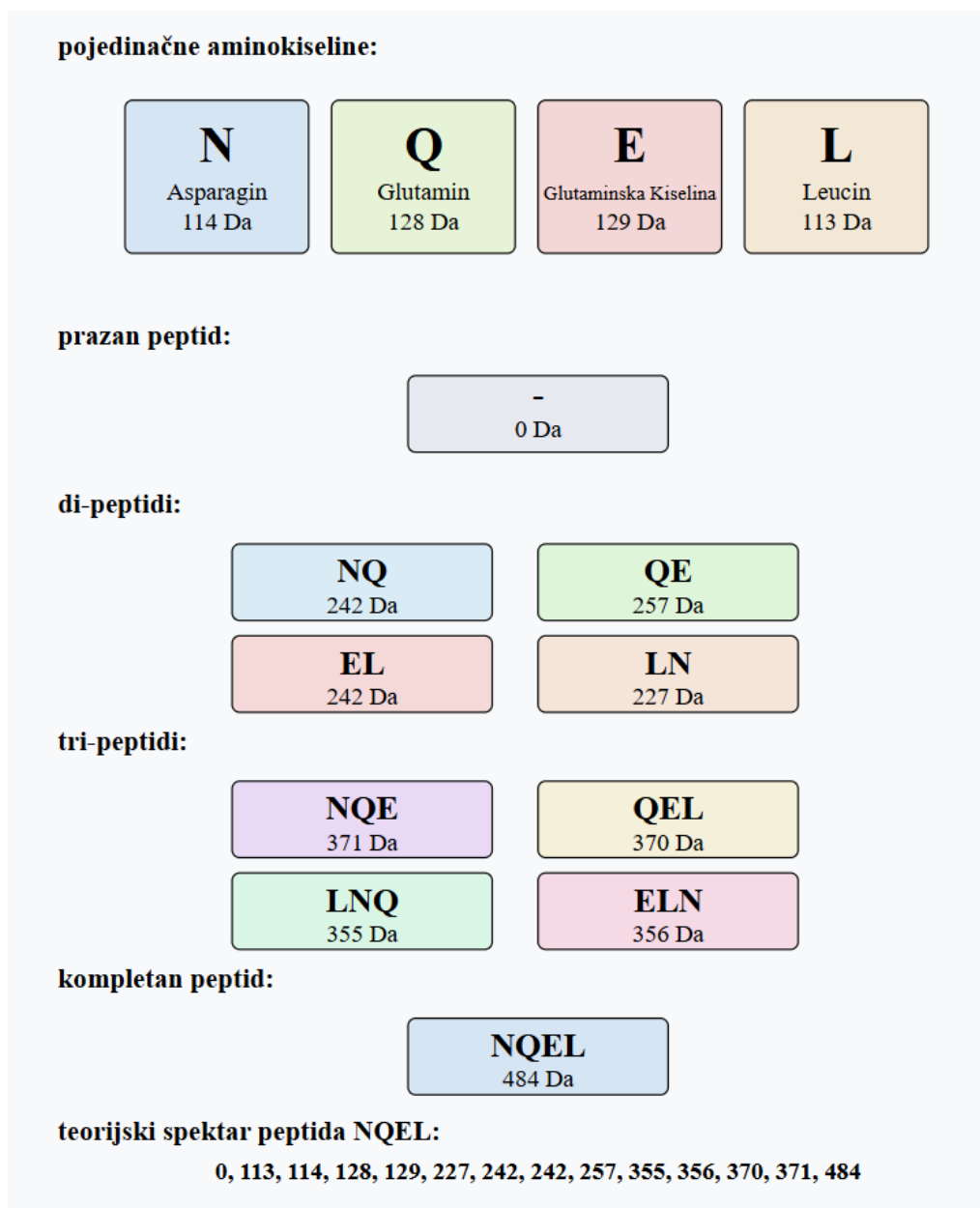
$$\begin{array}{cccccccccc}
 V & K & L & F & P & W & F & N & Q & Y \\
 99 & + & 128 & + & 113 & + & 147 & + & 97 & + & 186 & + & 147 & + & 114 & + & 128 & + & 163 & = & 1322
 \end{array}$$

2.4 Teorijski spektar peptida

Teorijski spektar peptida predstavlja mase svih mogućih potpeptida, uključujući 0 i masu celog peptida. Na osnovu peptida možemo lako da odredimo teorijski spektar ali na osnovu spektra ne možemo lako da odredimo koji je peptid u pitanju.

Problem sekvenciranja ciklopeptida samim tim se svodi na problem kako rekonstruisati ciklični peptid na osnovu njegovog teorijskog spektra. U nastavku će biti prikazano nekoliko različitih algoritama za sekvenciranje peptida.

Kao ulaz u svaki od ovih algoritama očekuje se eksperimentalni spektar, odnosno spektar koji je dobijen uz pomoć masenog spektrometra za neki peptid. Na Slici 2.4 su prikazane mase svih potpeptida peptida **NQEL** koje se dobijaju uz pomoć masenog spektrometra, kao i masa praznog peptida i celog peptida, takođe je prikazan i teorijski spektar.



Slika 2.4: Teorijski spektar peptida **NQEL** koji prikazuje sve moguće potpeptide, njihove mase i njegov teorijski spektar

Glava 3

Algoritmi za sekvenciranje

U ovom odeljku biće prikazani algoritmi za određivanje cikličnog peptida na osnovu poznatog eksperimentalnog spektra. Neki od algoritama koji će biti objašnjeni su:

- **Pristup grubom silom** (*Brute force*): Direktan pristup gde se isprobavaju sve moguće kombinacije da bi se našlo optimalno rešenje.
- **Branch and Bound**: Optimizovan algoritam koji će odbacivati kandidate čim prestanu da budu potencijalno rešenje.
- **Leaderboard algoritam**: Algoritam koji održava listu N najboljih kandidata za rešenje i na osnovu njih smanjuje broj potencijalnih kandidata.
- **Spektralna konvolucija**: Određuje aminokiseline koje mogu da učestvuju u peptidu na osnovu eksperimentalnog spektra.
- **DeepNovo**: Metoda zasnovana na dubokom učenju koja omogućava sekvenciranje peptida bez oslanjanja na baze podataka.

3.1 Pristup grubom silom (*Brute Force*)

Pristup grubom silom predstavlja osnovni pristup sekvenciranju koji sistematski ispituje sve moguće kombinacije aminokiselina dok ne pronađe sekvencu koja najbolje odgovara eksperimentalnom spektru [12, 14]. Iako jednostavan za implementaciju, ovaj pristup postaje neizvodiv za duže sekvence zbog eksponencijalnog rasta prostora pretrage.

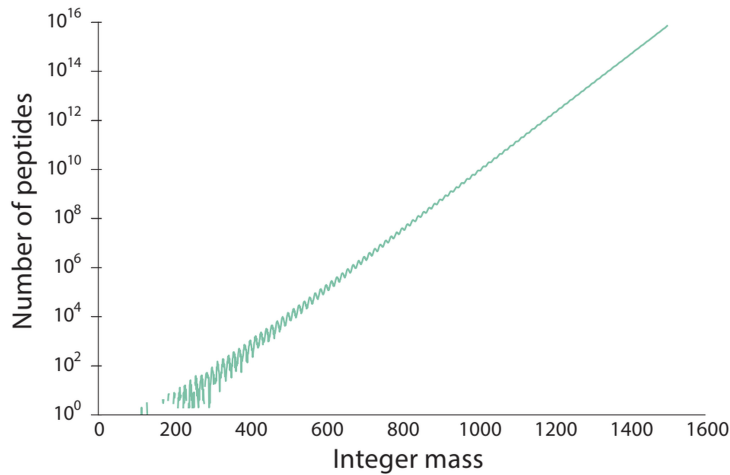
Na primer, za peptid mase 579 Da, algoritam će generisati sve moguće kombinacije aminokiselina i proveriti da li njihova ukupna masa odgovara zadatoj masi. Kao što je prikazano u Tabeli 3.1, mogu postojati različiti peptidi sa istom ukupnom masom (**FPAYT** i **QNWGS**), što dodatno komplikuje problem.

F	147 Da	Q	128 Da
P	97 Da	N	114 Da
A	71 Da	W	186 Da
Y	163 Da	G	57 Da
T	101 Da	S	94 Da
Ukupna masa: 579 Da		Ukupna masa: 579 Da	

Tabela 3.1: Poređenje aminokiselina i njihove mase

Da bi se utvrdilo koji od peptida je tačno rešenje, algoritam mora da generiše teorijski spektar za svaki kandidat peptid i uporedi ga sa eksperimentalnim spektrom. Ovo dodatno povećava računsku složenost algoritma, ali je neophodno za pronalaženje tačnog rešenja.

Na Slici 3.1, možemo da vidimo koliko postoji različitih peptida za istu masu, samim tim možemo zaključiti da je izvršavanje pristupom grubom silom veoma neefikasno.



Slika 3.1: Broj peptida koji imaju istu masu. Preuzeto iz [14].

Na osnovu prethodnog teksta definiše se pseudokod za pristup grubom silom koji se može videti kao algoritam 1.

Algoritam 1: Pristup grubom silom

Funkcija GrubaSila(*eksperimentalniSpektar*)

```
    peptidi  $\leftarrow$  Lista sa praznim stringom
    rezultati  $\leftarrow$  Prazna lista
    ciljna_masa  $\leftarrow$  Poslednji element eksperimentalniSpektar
    while peptidi  $\neq$  Prazno do
        prosireni  $\leftarrow$  Proširi(peptidi)
        kandidati  $\leftarrow$  Prazna lista
        foreach peptid  $\in$  prosireni do
            masa  $\leftarrow$  IzračunajMasu(peptid)
            if masa = ciljna_masa then
                if CikličniSpektar(peptid) = eksperimentalniSpektar then
                    Dodaj peptid u rezultati
                end
            else if masa < ciljna_masa then
                Dodaj peptid u kandidati
            end
        end
        peptidi  $\leftarrow$  kandidati
    end
    return rezultati
```

Objašnjenje pomoćnih funkcija:

- **Proširi(peptidi)** – sve preostale peptide proširuje sa svakom mogućom aminokiselinom i vraća proširenu listu.
- **IzračunajMasu(peptid)** – računa ukupnu masu peptida sabiranjem masa svih aminokiselina koje čine taj peptid.
- **CikličniSpektar(peptid)** – za peptid koji je potencijalno rešenje generiše se ciklični spektar koji se poredi sa zadatim eksperimentalnim spektrom.

3.2 Branch and Bound

Branch and Bound algoritam [12, 14] je optimizovana verzija pristupa grubom silom koja koristi strategiju „podeli pa vladaj“ za efikasnije pretraživanje prostora

rešenja. Za razliku od pristupa grubom silom koji ispituje sve moguće kombinacije, *Branch and Bound* algoritam inteligentno eliminiše delove prostora pretrage koji ne mogu sadržati optimalno rešenje.

U kontekstu sekvenciranja peptida, algoritam funkcioniše na sledeći način:

- **Grananje (*Branch*):** Algoritam gradi stablo pretrage gde svaki čvor predstavlja delimičnu sekvencu peptida. Svaki čvor se grana dodavanjem nove aminokiseline na postojeću sekvencu.
- **Ograničavanje (*Bound*):** Za svaki čvor, algoritam procenjuje da li taj put može dovesti do validnog rešenja. Ako masa peptida već premašuje ciljanu masu ili ako teorijski spektar delimične sekvence nije u skladu sa eksperimentalnim, ta grana se odseca i dalje se ne istražuje.
- **Optimizacija:** Algoritam može koristiti dodatne heuristike za procenu koje grane prvo istražiti, što dodatno ubrzava pronalaženje rešenja.

Prednosti *Branch and Bound* algoritma u odnosu na pristup grubom silom su značajne i njihovo poređenje može da se vidi u Tabeli 3.2.

Pristup grubom silom	Branch and Bound
Istražuje sve moguće kombinacije	Inteligentno eliminiše neperspektivne grane
Eksplozivna vremenska složenost	Značajno bolja vremenska složenost (u najgorem slučaju i dalje eksplozivna, ali znatno brža)
Garantuje pronalaženje svih rešenja	I dalje garantuje pronalaženje svih rešenja
Neefikasan za duže peptide	Efikasniji za duže peptide

Tabela 3.2: Poređenje pristupa grubom silom i *Branch and Bound* pristupa

Pseudokod za algoritam Branch and Bound može se videti kao algoritam 2.

Algoritam 2: Branch and Bound

Funkcija BranchAndBound(*eksperimentalniSpektar*)

```
    peptidi  $\leftarrow$  Lista sa praznim stringom  
    rezultati  $\leftarrow$  Prazna lista  
    ciljna_masa  $\leftarrow$  Poslednji element eksperimentalniSpektar  
    while peptidi  $\neq$  Prazno do  
        | prosireni  $\leftarrow$  Proširi(peptidi)  
        | kandidati  $\leftarrow$  Prazna lista  
        | foreach peptid  $\in$  prosireni do  
            | masa  $\leftarrow$  IzračunajMasu(peptid)  
            | if masa = ciljna_masa then  
                | if CikličniSpektar(peptid) = eksperimentalniSpektar then  
                    | Dodaj peptid u rezultati  
                | end  
            | else if masa < ciljna_masa then  
                | if Konzistentan(peptid, eksperimentalniSpektar) then  
                    | Dodaj peptid u kandidati  
                | end  
            | end  
        | end  
        | peptidi  $\leftarrow$  kandidati  
    end  
    return rezultati
```

Objašnjenje pomoćnih funkcija:

- **Konzistentan**(*peptid*, *eksperimentalniSpektar*) – glavni korak odsecanja i smanjivanja prostora pretrage. Provera da li se svaka masa u okviru linearnog spektra peptida javlja i u okviru eksperimentalnog spektra. Ako neka masa postoji u okviru linearnog spektra a ne u okviru eksperimentalnog spektra to znaci da spektrum nije konzistentan, obrnuto ne mora da važi, jer se računa spektar parcijalnog peptida a ne još celog pa masa koja trenutno nije prisutna može da se pojavi.

3.3 Uporedna analiza prethodnih algoritama

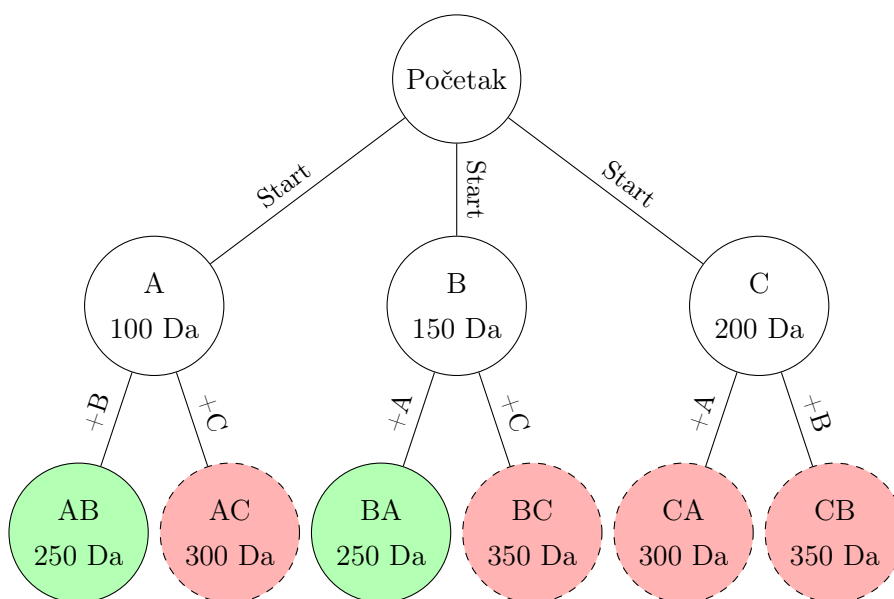
Posmatramo primer sa 3 aminokiseline:

- A (masa = 100 Da)
- B (masa = 150 Da)
- C (masa = 200 Da)

Tražimo sekvencu AB (ukupna masa = 250 Da). Poredimo pristup grubom silom i pristup grananja i ograničavanja.

3.3.1 Pristup grubom silom

Ispituje sve kombinacije i odbacuje samo kada ukupna masa premaši traženu masu:

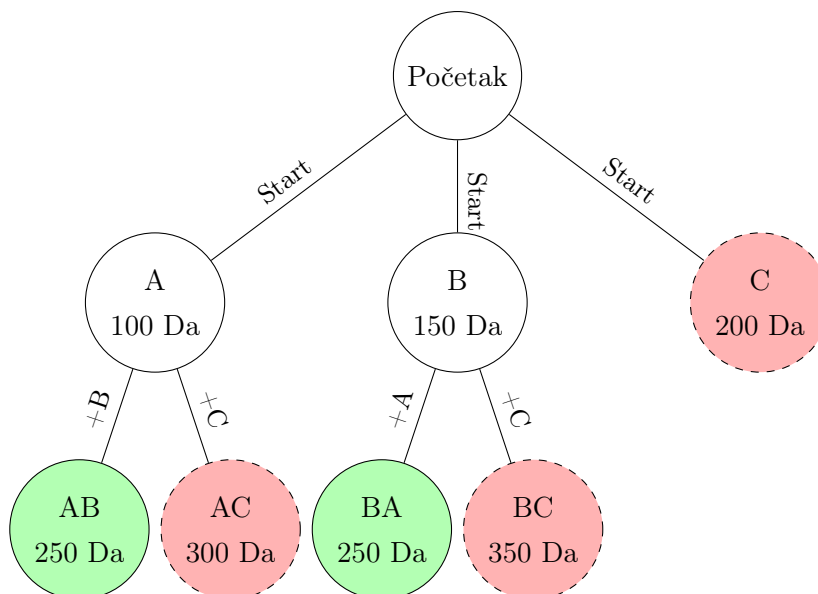


Objašnjenje:

- **Zeleno** - pronađena rešenja (AB, BA)
- **Crveno** - odbačene grane (masa > 250 Da)
- Ukupno evaluacija: 3 (prvi nivo) + 6 (drugi nivo) = 9 čvorova

3.3.2 Algoritam *Branch and Bound*

Odbacuje grane gde bilo koji deo sekvence nema masu koja postoji u eksperimentalnom spektru:



Objašnjenje:

- **Zeleno** - pronađena rešenja (AB, BA)
- **Crveno** - odbačene grane (C nema masu iz spektra)
- Ukupno evaluacija: 3 (prvi nivo) + 4 (drugi nivo) = 7 čvorova

3.3.3 Zaključak

Algoritam grananja i ograničavanja značajno smanjuje prostor pretrage eliminacijom neperspektivnih grana već u ranim fazama, dok gruba sila mora da ispita sve kombinacije. Na ovom primeru može se primetiti kako se odmah odbacuje cela **C** grana i svi njeni potomci. U ovom slučaju smanjuje se prostor pretrage za samo dva čvora, ali u stvarnosti postoji 20 aminokiselina i odsecanje čvorova i njihovih potomaka u samom startu predstavlja ogromno ubrzanje i poboljšanje u odnosu na pristup grubom silom.

3.4 Nedostak prethodnih algoritama

Glavni nedostatak prethodnih algoritama je to što teorijski spektar peptida koji je kandidat za rešenje mora skroz da se poklopi sa zadatim eksperimentalnim spektrom. U realnosti to nije moguće, eksperimentalni spektri često sadrže lažne ili nedostajuće mase i samim tim mase neće biti iste kao u teorijskom spektru. **Lažna masa** predstavlja masu koja se nalazi u eksperimentalnom spektru ali zapravo ne postoji u teorijskom spektru peptida. **Nedostajuća masa** predstavlja masu koja se ne nalazi u eksperimentalnom spektru ali postoji u teorijskom spektru peptida.

U Tabeli 3.3 može da se vidi primer teorijskog i eksperimentalnog spektra za peptid **NEQ** sa nedostajućim i lažnim masama.

eksperimentalni	0	114	128		133	200		243		371
teorijski	0	114	128	129			242	243	257	371

Tabela 3.3: Prikaz nedostajućih masa (obojene zelenom bojom) i lažnih masa (obojene plavom bojom) koje mogu da se jave u eksperimentalnom spektru za peptid **NQE**

U narednim sekcijama dat je prikaz algoritama koji su dizajnirani tako da rade sa lažnim i nedostajućim masama.

3.5 *Leaderboard* algoritam

Leaderboard algoritam [12, 14] predstavlja optimizovani pristup za sekvenciranje peptida. Za razliku od sekvenciranja grubom silom koje zahteva tačno poklapanje između teorijskog spektra kandidata i eksperimentalnog spektra, ovaj algoritam je dizajniran da radi sa nedostajućim i lažnim masama tako što prati samo najbolje kandidate peptida umesto svih mogućnosti. *Leaderboard* algoritam održava listu najboljih kandidata tokom pretrage. U svakoj iteraciji, algoritam proširuje sekvence sa svim mogućim aminokiselinama i zadržava one kandidate čiji teorijski spektar linearnog peptida ima najveći broj poklapanja sa zadatim eksperimentalnim spektrom. Određuje se teorijski spektar linearnog peptida zato što peptidi još nisu do kraja formirani tako da se još ne zna kako bi oni izgledali ciklično.

3.5.1 Prednosti algoritma

- **Efikasnost:** Fokusira se samo na najperspektivnije kandidate, značajno smanjujući vreme izvršavanja.

- **Skalabilnost:** Efikasno radi sa peptidima različitih dužina bez eksponencijalnog rasta vremena.
- **Preciznost:** Održava visoku tačnost uprkos šumu u eksperimentalnim podacima.

3.5.2 Primene

- **Eksperimentalni podaci:** Idealan za analizu realnih podataka masene spektrometrije sa šumom. Pokazao se kao dobar algoritam koji može da pronađe rešenje i kada je broj lažnih ili nedostajućih masa iz spektra 10%, u slučaju kada se taj broj poveća na 25%, ovaj algoritam ne nalazi uvek tačno rešenje.
- **Nepotpuni podaci:** Kada tačno poklapanje nije moguće zbog nepotpunih ili netačnih podataka u spektru.
- **Vremenski kritične analize:** Kada je potrebna brza identifikacija peptida iz velikih skupova podataka.

Algoritam 3 prikazuje pseudokod *Leaderboard* algoritma.

Algoritam 3: Leaderboard sekvenciranje

```
Funkcija Leaderboard(eksperimentalniSpektar)  
    peptidi  $\leftarrow$  Lista sa praznim stringom  
    leader_peptid  $\leftarrow$  Prazan peptid  
    najbolji_rezultat  $\leftarrow$  0  
    ciljna_masa  $\leftarrow$  Poslednji element eksperimentalniSpektar  
    while peptidi  $\neq$  Prazno do  
        prosireni  $\leftarrow$  Proširi(peptidi)  
        kandidati  $\leftarrow$  Prazna lista  
        foreach peptid  $\in$  prosireni do  
            masa  $\leftarrow$  IzračunajMasu(peptid)  
            if masa = ciljna_masa then  
                rezultat_kandidata  $\leftarrow$  CikličniScore(peptid,  
                    eksperimentalniSpektar)  
                if rezultat_kandidata > najbolji_rezultat then  
                    leader_peptid  $\leftarrow$  peptid  
                    najbolji_rezultat  $\leftarrow$  rezultat  
                end  
            else if masa < ciljna_masa then  
                Dodaj peptid u kandidati  
            end  
        end  
    peptidi  $\leftarrow$  Trim(kandidati, eksperimentalniSpektar, N)  
    end  
    return leader_peptid
```

Objašnjenje pomoćnih funkcija:

- **Trim(kandidati, eksperimentalniSpektar, N)** – Jedna od glavnih funkcija. Ulaz u funkciju predstavljaju peptidi koji su kandidati za rešenje, eksperimentalni spektar kao i broj peptida koji ćemo vratiti iz funkcije odnosno najbolji kandidati za potencijalno rešenje. Bitno je izabrati dobro broj kandidata koji prolazi u dalju rundu. U slučaju da je taj broj previše mali rizujemo da previše agresivno odsečemo neke kandidate i da potencijalno izgubimo rešenje. U slučaju da je broj previše veliki čuvaćemo previše kandidata i samim tim povećati vreme izvršavanja algoritma. Generalno, dobra je praksa ako se traže peptidi sa manjom masom da se koristi manji broj kandidata koji nastavlja u

sledeću rundu a ako se masa poveća da se samim tim poveća i broj kandidata koji nastavlja u sledeću rundu. Funkcija **Trim** koristi funkciju **linearScore** koja računa broj poklapanja teorijskog spektra peptida sa eksperimentalnim spektrom. Ova funkcija se koristi kada se ceo peptid još ne zna i samim tim ne mogu da se kreiraju sve ciklične varijacije jer bi se dobile mase koje se možda ne bi dobile kada se peptid proširi aminokiselinama. Na kraju **Trim** kandidati se sortiraju opadajuće po linearnom skor i u sledeću rundu prolaze prvih **N** kandidata kao i svi kandidati koji imaju isti rezultat kao kandidat na poziciji **N**. Na ovaj način osiguravamo da sa sigurnošću svi dobri kandidati prođu u sledeću rundu.

- **CikličniScore(peptid, eksperimentalniSpektar)** – Računa broj poklapanja teorijskog spektra cikličnog peptida sa eksperimentalnim spektrom. Ova funkcija se koristi u slučaju da je masa peptida jednaka najvećoj teorijskoj masi jer je u tom slučaju formiran ceo peptid i mogu da se nađu svi potpeptidi.

3.6 Spektralna konvolucija

Spektralna konvolucija [12, 14] je tehnika koja se koristi za identifikaciju aminokiselina koje mogu biti prisutne u peptidu na osnovu eksperimentalnog spektra. Ova metoda analizira razlike između masa u spektru i identifikuje one koje odgovaraju masama aminokiselina.

Proces se sastoji iz dva glavna koraka:

- **Izračunavanje konvolucije:** Za svaki par masa u spektru, izračunava se njihova razlika. Ove razlike mogu odgovarati masama aminokiselina. Pravi se matrica konvolucije, koja predstavlja donjetrougaonu matricu gde je prva vrsta i prva kolona eksperimentalni spektar a pozicije u matrici predstavljaju apsolutnu vrednost razlike elemenata sa tim indeksom iz spektra.
- **Identifikacija aminokiselina:** Najčešće razlike koje se pojavljuju u spektru verovatno odgovaraju aminokiselinama prisutnim u peptidu. Te aminokiseline se izdvajaju i koriste u *Leaderboard* algoritmu.

Glavna prednost ovog algoritma jeste to što u samom startu smanjuje skup aminokiselina koje mogu da učestvuju u građenju peptida, čime se algoritam dosta

ubrzava. Takođe, ovim se otvara mogućnost da se identifikuju nepoznate ili modifikovane aminokiseline. Još jedna od prednosti ovog algoritma jeste to što može da radi na eksperimentalnim spektrima koji imaju još više pogrešnih ili nedostajućih masa.

Algoritam 4 prikazuje pseudokod algoritma spektralne konvolucije.

Algoritam 4: Spektralna konvolucija

```

Funkcija SpektralnaKonvolucija(eksperimentalniSpektar)
    matrica_konvolucije  $\leftarrow$  Prazna lista
    n  $\leftarrow$  Broj elemenata u eksperimentalniSpektar
    for i  $\leftarrow$  0 to n - 1 do
        for j  $\leftarrow$  0 to i - 1 do
            masa  $\leftarrow$  S[i] - S[j]
            if  $57 \leq \textit{masa} \leq 200$  then
                Dodaj masa u konvolucija
            end
        end
    end
    broj_pojavljivanja  $\leftarrow$  Prazna mapa
    foreach masa u konvolucija do
        if masa  $\in$  broj_pojavljivanja then
            broj_pojavljivanja[masa]  $\leftarrow$  broj_pojavljivanja[masa] + 1
        end
        else
            broj_pojavljivanja[masa]  $\leftarrow$  1
        end
    end
    sortirane_frekvencije  $\leftarrow$  SortirajOpadajuće(broj_pojavljivanja)
    najcesce_mase  $\leftarrow$  Uzimamo S najčešćih masa iz
        sortirane_frekvencije
    leader_peptid  $\leftarrow$  LeaderboardSequencing(eksperimentalniSpektar,
        najcesce_mase)
    return leader_peptid

```

Objašnjenje pomoćnih funkcija:

- **SortirajOpadajuće**(*broj_pojavljivanja*) – U konkretnoj implementaciji matrica konvolucije je najlakše da bude lista da bi se što lakše iteriralo kroz

nju. Na osnovu matrice se formiraju frekvencije masa i sortira se opadajuće na osnovu broja pojavljivanja masa.

- **LeaderboardSequencing(eksperimentalniSpektar, najcesce_mase)** – Poziva se prethodno implementiran *Leaderboard* algoritam, samo se ne koriste sve aminokiseline nego se koristi smanjen skup aminokiselina za proširivanje i pravljenje novih peptida.

U Tabeli 3.4 može da se vidi matrica konvolucije za eksperimentalni spektar:

0 114 128 129 242 243 257 371

	Mase (Da)						
	0	114	128	129	242	243	257
0	–	–	–	–	–	–	–
114	114	–	–	–	–	–	–
128	128	14	–	–	–	–	–
129	129	15	1	–	–	–	–
242	242	128	114	113	–	–	–
243	243	129	115	114	1	–	–
257	257	143	129	128	15	14	–
371	371	257	243	242	129	128	114

Tabela 3.4: Matrica konvolucije

Na osnovu Tabele 3.4 možemo da vidimo koje su to mase koje se najviše puta ponavljaju. Vidimo da se mase 114, 128 i 129 pojavljuju 4 puta i njih ćemo sigurno koristiti u *Leaderboard* algoritmu, preostale mase koje bi se koristile zavise od broja **S**, koji nam govori koliko najčešćih masa ćemo uzeti.

Na osnovu Tabele 1.1 možemo da vidimo da masa 114 odgovara aminokiselini sa skraćenicom **N**, da masa 128 odgovara **K** i **Q** aminokiselinama a da masa 129 odgovara aminokiselini **E**. Rešenja zadatog teorijskog spektra su peptidi **NQE** i **NKE** i njihove ciklične kombinacije. Možemo da primetimo da smo u samom startu suzili izbor aminokiselina sa 20 na samo 3 koje predstavljaju rešenje, čime smo u velikoj meri ubrzali proces pronalaska peptida.

3.7 *DeepNovo*

Dva osnovna pristupa koja se koriste prilikom sekvenciranja peptida su pristup pretraživanja baze podataka i *de novo* pristup.

Pristup pretraživanja baze podataka se oslanja se na poređenje eksperimentalnih podataka sa bazom podataka poznatih proteinskih sekvenci, ali neki od problema u ovom pristupu su sledeći:

- **nepoznati proteini** - Novi proteini koji nikada ranije nisu viđeni neće se podudarati ni sa čim u bazi podataka, što dovodi do nemogućnosti identifikacije.
- **problem sa podacima** - Mase dobijene masenim spektrometrom mogu biti lažne i nedostajuće, što otežava pouzdano poređenje.

Pristup *de novo* sekvenciranja pokušava izgraditi sekvencu peptida iz početka, bez oslanjanja na bazu podataka i koristeći samo podatke koji su dobijeni masenom spektrometrijom. U okviru ovog pristupa koriste se i različiti algoritmi, kao što su pristup grubom silom, *Branch and Bound*, *Leaderboard* algoritam, kao i spektralna konvolucija, koji pokušavaju da generišu sekvence peptida i da porede njihove teorijske spektre sa eksperimentalnim. Međutim, iako koristan za identifikaciju novih peptida, *de novo* pristup je često manje precizan i računski zahtevan [17].

Trenutne tehnike za sekvenciranje peptida (poput pretraživanja baze podataka, *de novo* sekvenciranja, kao i raznih algoritamskih pristupa) mogu imati poteškoća u radu sa novim, složenim ili nepotpunim podacima [17]. *DeepNovo* kombinuje prednosti oba pristupa koristeći duboko učenje za poboljšanje tačnosti *de novo* sekvenciranja.

3.7.1 Računska složenost

De novo sekvenciranje, koje pokušava rekonstruisati sekvencu peptida bez oslanjanja na bazu podataka, suočava se sa značajnim računskim izazovima:

- Eksponencijalni rast prostora pretrage sa povećanjem dužine peptida.
- Potreba za složenim algoritmima za interpretaciju spektralnih podataka.
- Teškoće u razlikovanju izobaričnih aminokiselina (aminokiseline sa istom ili vrlo sličnom masom) - primeri takvih aminokiselina su *I* i *L* čija je masa 113

daltona, kao i K i Q čija je zaokružena masa 128 daltona dok su njihove tačne mase 128.094963 Da (K) i 128.058578 Da (Q), pa je njihova razlika 0.036385 Da.

3.7.2 Tehnike zasnovane na *De Novo* sekvenciranju

Pored *DeepNovo* tehnike koja će biti opisana u ovom radu, postoje i još neke tehnike zasnovane na *De Novo* principu:

- **PEAKS** [16] - koristi usmerene aciklične grafove
- **Novor** [13] - koristi klasifikacione modele mašinskog učenja da odredi sekvencu aminokiselina sa najvećom verovatnoćom
- **PepNovo** [10] - koristi modelovanje verovatnoća pomoću grafova

DeepNovo je dizajniran da prevazilazi računske izazove koristeći moć dubokog učenja a rezultati će pokazati da je bolji i od drugih metoda koje su zasnovane na *de novo* principu.

3.7.3 Opšti pregled

DeepNovo [17] je metoda zasnovana na dubokom učenju koja poboljšava sekvenciranje peptida koristeći algoritam za predviđanje sekvenci aminokiselina iz podataka generisanih masenom spektrometrijom. Osnovna ideja je da model dubokog učenja može naučiti obrasce u podacima masene spektrometrije i predvideti sekvencu peptida bez potrebe za oslanjanjem na referentnu bazu podataka. Ova inovativna metoda pokazuje značajno poboljšanje u tačnosti sekvenciranja, posebno u slučajevima kada su peptidne sekvence nove i ne podudaraju se ni sa jednom već poznatom.

Proces treniranja

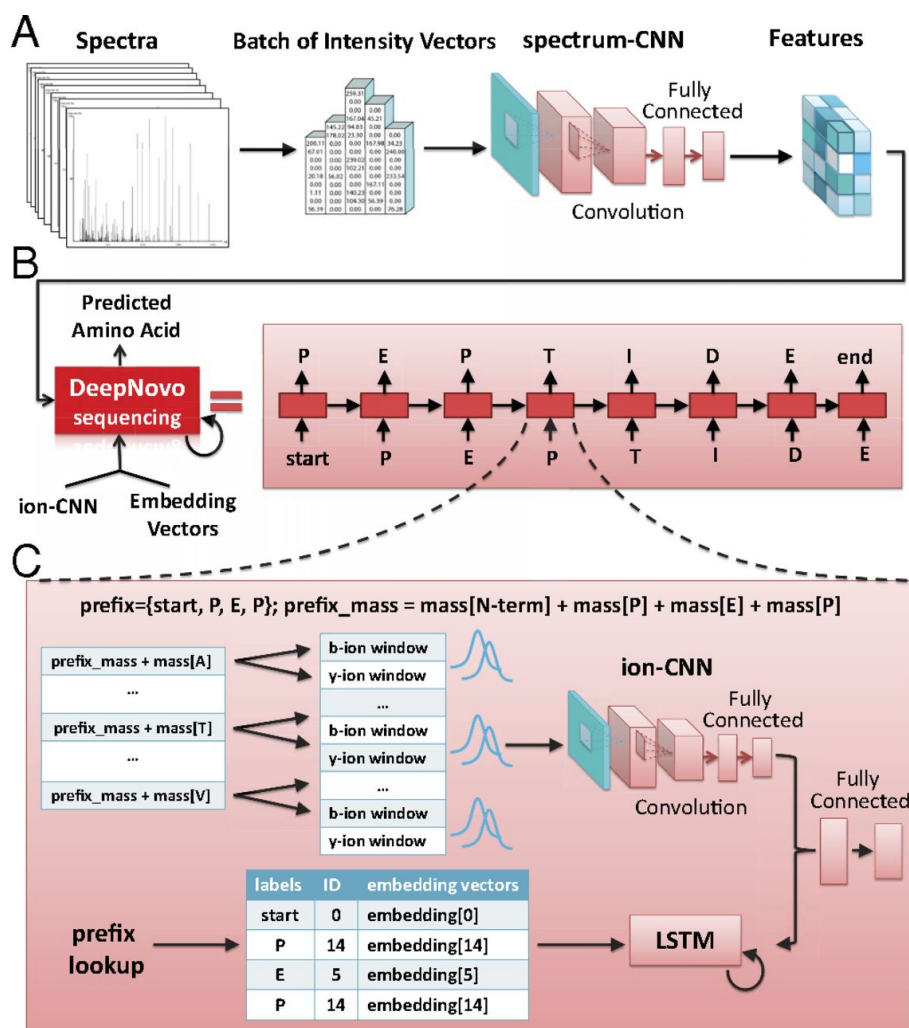
DeepNovo se trenira na velikom skupu podataka poznatih sekvenci peptida i njihovih odgovarajućih podataka masene spektrometrije. Model uči da preslikava eksperimentalne podatke na sekvence peptida kroz proces nadgledanog učenja, gde se optimizuju parametri mreže da bi se minimizirala razlika između predviđenih i stvarnih sekvenci.

Proces treniranja obuhvata sledeće ključne faze:

1. Prikupljanje peptidnih sekvenci i njihovih spektara
2. Pretprocesiranje spektralnih podataka za normalizaciju i uklanjanje šuma
3. Treniranje neuronske mreže da predviđa aminokiseline na osnovu spektralnih karakteristika
4. Optimizacija modela korišćenjem tehnika kao što su regularizacija i rano zaustavljanje
5. Validacija modela na nezavisnom skupu podataka za procenu performansi

3.7.4 Arhitektura neuronske mreže

DeepNovo koristi hibridnu arhitekturu koja kombinuje konvolutivne i rekurentne neuronske mreže i može se videti na Slici 3.2.



Slika 3.2: Arhitektura DeepNovo pristupa, preuzeto sa [17]

Konvolutivne neuronske mreže (CNN)

CNN se koristi za otkrivanje šablona u ulaznim podacima. Oslanja se na tehniku *sliding window* i procesuiru male lokalne regione koristeći filtere. Ova mreža je trenirana da prepozna različite tipove jona i da pretvori sirove podatke u reprezentaciju svojstava ulaznih podataka.

U ovom modelu su korišćene dve CNN mreže:

- **Spektralna CNN**: Kao deo preprocesiranja podataka spektar dobijen masenom spektrometrijom transformiše se u vektor fiksne dužine (npr. 50 000 elemenata) i dobijeni vektor se prosleđuje ovoj mreži. Na ovaj način, CNN uči šablone nad svim podacima spektra i kao rezultat ove mreže dobija se novi

vektor koji se dalje koristi u rekurentnoj mreži. Dobijeni vektor predstavlja karakteristike koje je *CNN* mreža, uz pomoć filtera, naučila nad podacima masenog spektrometra. Konvolutivna mreža se sastoji od 3 konvolutivna sloja i koristi *ReLU* aktivacionu funkciju. Ovo predstavlja značajan deo arhitekture jer može dosta da poboljša preciznost sekvenciranja i može da nauči šablone u spektru čak i ako su neki maksimumi u spektru pomereni zbog šuma.

- **Jonska CNN:** Ova mreža se koristi tokom odabira sledeće aminokiseline u peptidu i služi da iz malog segmenta spektra izdvoji najvažnije informacije. Za svakog kandidata aminokiseline, mreža analizira deo spektra i procenjuje da li se očekivani jonski fragmenti pojavljuju na odgovarajućim mestima. Prilikom svakog koraka predviđanja sledeće aminokiseline *DeepNovo* koristi dosadašnje predikcije da izračuna prefiksnu masu, na osnovu koje generiše parcijalni teorijski spektar i proverava da li eksperimentalni spektar podržava ovaj izbor. Ovaj deo modela je bitan u slučaju da su podaci šumoviti ili da neke mase nedostaju u eksperimentalnom spektru.

Rekurentne neuronske mreže (RNN)

Ove mreže su dizajnirane za predviđanje sekvenci, gde predikcija sledećeg elementa (aminokiseline) ne zavisi samo od prethodno predviđenih aminokiselina, već i od karakteristika spektra koje opisuju ceo peptid. U kontekstu sekvenciranja peptida, **RNN** može naučiti kako izbor jedne aminokiseline utiče na pojavljivanje sledeće u sekvenci, što je ključno za tačno predviđanje.

DeepNovo koristi posebnu vrstu RNN-a koja se zove *Long Short-Term Memory (LSTM)*. Ključna prednost **LSTM** mreža je ta što bolje prate nesusedne zavisnosti, što znači da peptidi mogu da budu različitih dužina a ova mreža pamti i odnose koji su udaljeni. Ovo je bitno jer početak sekvence može da utiče na predviđanje neke kasnije aminokiseline.

Ova mreža se sastoji od 1 sloja i radi tako što dodaje jednu po jednu aminokiselinu sve dok ne stigne do kraja peptida. U svakom koraku **LSTM** mreža uzima u obzir:

- Šta je mreža naučila do sada - trenutno stanje
- Sledeća aminokiselina koja je kandidat
- Svojstva spektra koja su dobijana od konvolutivne mreže

Kao izlaz iz ove mreže koristi se *softmax* projekcija i ona određuje za svaku aminokiselinu koja je verovatnoća da se ona nalazi na sledećoj poziciji u sekvenci. Dodatno, koristi se *beam search*, odnosno ne bira se samo aminokiselina sa najvećom verovatnoćom nego se čuva više kandidata koji imaju veću verovatnoću. Ovim postupkom se povećava preciznost i gledaju se alternativna rešenja. Na kraju se sekvence rangiraju po skor koji pokazuje koliko se poklapaju sa traženim spektrom i koliko imaju grešaka i bira se najbolja moguća.

3.7.5 Rezultati i evaluacija

DeepNovo metoda nadmašuje tradicionalne metode, posebno u slučajevima kada su sekvence peptida nove i ne podudaraju se ni sa jednom poznatom proteinskom bazom podataka.

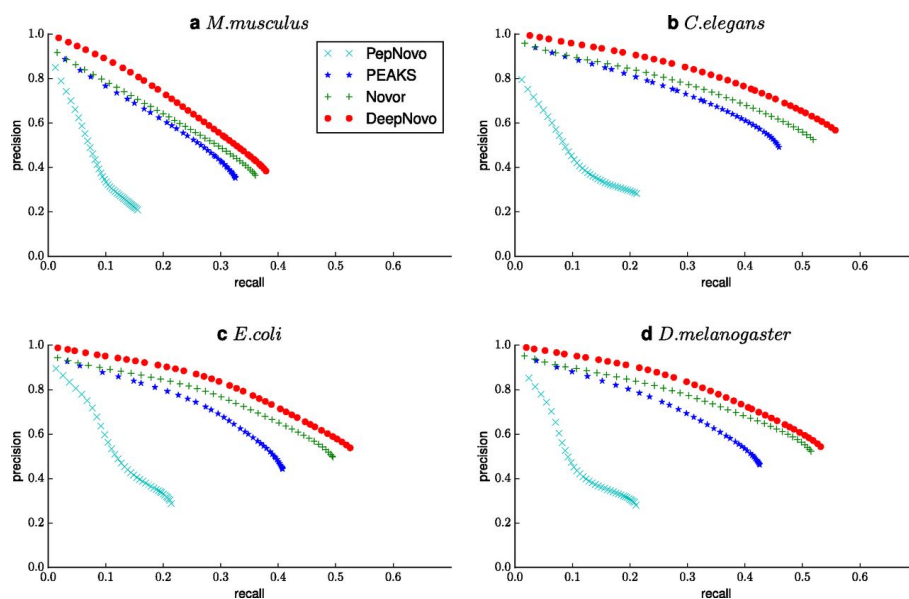
Eksperimenti su pokazali značajno poboljšanje u tačnosti identifikacije aminokiselina, posebno za složene peptide. Za potrebe testiranja naučnici su koristili podatke različitih vrsta. Na Slici 3.3 su prikazani rezultati poređenja više algoritama. Da bi se merila preciznost rešenja poređena je prava sekvenca aminokiselina sa onom koja je dobijena na osnovu spektra. Takođe, koristile su se različite metrike:

- **Preciznost (eng. precision)** - predstavlja odnos broja peptida koje je generisao algoritam koji su zapravo tačni i ukupnog broja peptida koje je generisao
- **Odziv (eng. recall)** - predstavlja odnos broja tačnih peptida koje je generisao i koliko je ukupno bilo peptida koji su se tražili
- **AUC-PR** - predstavlja koliko dobro model balansira preciznost i odziv, što je veća vrednost bolje će biti i preformanse

Može se primetiti da je *DeepNovo* model na svakom od datih skupa podataka imao bolje rezultate. *DeepNovo* je imao i veću preciznost u traženju peptida kao i veći odziv, samim tim i odnos *AUC-PR* krive je bolji nego kod konkurenata, što znači da predlaže ispravne peptide i pronalazi većinu peptida.

DeepNovo je uspešno primenjen u nekoliko realnih scenarija:

- **Identifikacija novih antimikrobnih peptida:** Otkrivanje potencijalnih kandidata za nove antibiotike



Slika 3.3: Poređenje performansi DeepNovo sa drugim algoritmima, preuzeto sa [17]

- **Analiza post-translacionih modifikacija:** Detekcija peptida sa složenim modifikacijama koje su se desile nakon njegovog kreiranja, što može dovesti do neočekivanih masa koje je teško pronaći
- **Univerzalna primena:** Uspešna implementacija na različitim vrstama i organizmima

Glava 4

Elektronska platforma

U ovom poglavlju će biti opisana elektronska platforma koja je razvijena kao deo ovog rada. Biće objašnjeno njeno pokretanje i korišćenje kao i tehnologije koje su upotrebljene prilikom pravljenja platforme.

Frontend aplikacije je pisan u programskom jeziku **TypeScript** [9] uz korišćenje **Node.js 18** [7] i **Next.js framework**-a [6]. *Backend* aplikacije je implementiran u programskom jeziku **Python 3.12** [8] uz korišćenje **Django framework**-a [1]. Izvorni kod aplikacije se nalazi na *GitHub*-u, u javnom repozitorijumu [4].

4.1 Pokretanje aplikacije

Pokretanje aplikacije može da se odradi na nekoliko načina:

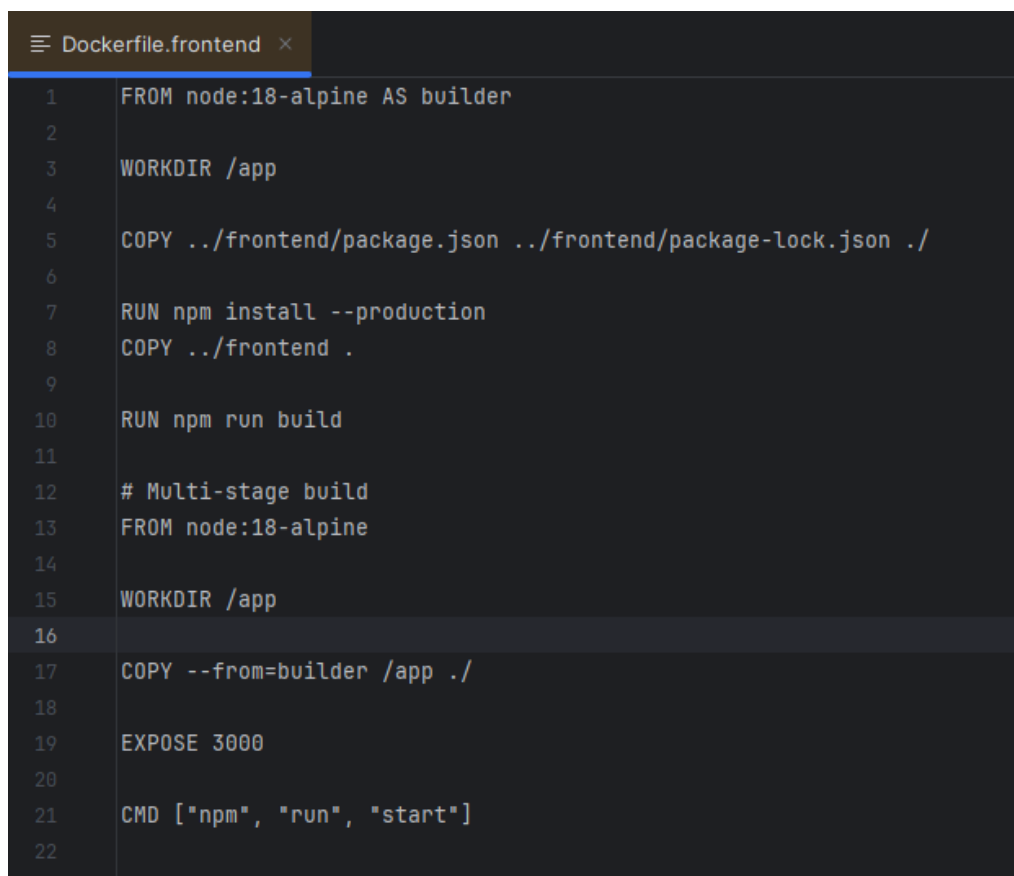
- **Docker Compose** [2] - najjednostavniji način pokretanja aplikacije uz korišćenje *Docker* alata [3]
- **Direktno pokretanje komponenti** - pokretanje posebno *Frontend* i posebno *Backend* komponenti
- **Google Cloud Run (GCR)** [5] - ova aplikacija je dostupna za korišćenje preko *Google Cloud Run* platforme, pa će biti objašnjeno i kako odraditi *deploy* aplikacije na **GCR**

4.1.1 *Docker Compose* konfiguracija

Za potrebe što lakšeg pokretanja aplikacije korišćen je *open source* alat *Docker* i *Docker Compose*. *Docker Compose* nam pruža jednostavan način da aplikaciju po-

krenemo sa svim definisanim bibliotekama i promenljivim okruženja bez potrebe da se bilo šta dodatno podešava. Ovo je posebno bitno u situacijama kada se aplikacija sastoji iz više komponenti pa je potrebno da se pokrene više kontejnera kao što je ovde slučaj. Unutar **docker** foldera nalaze *docker* i *docker-compose* fajlovi.

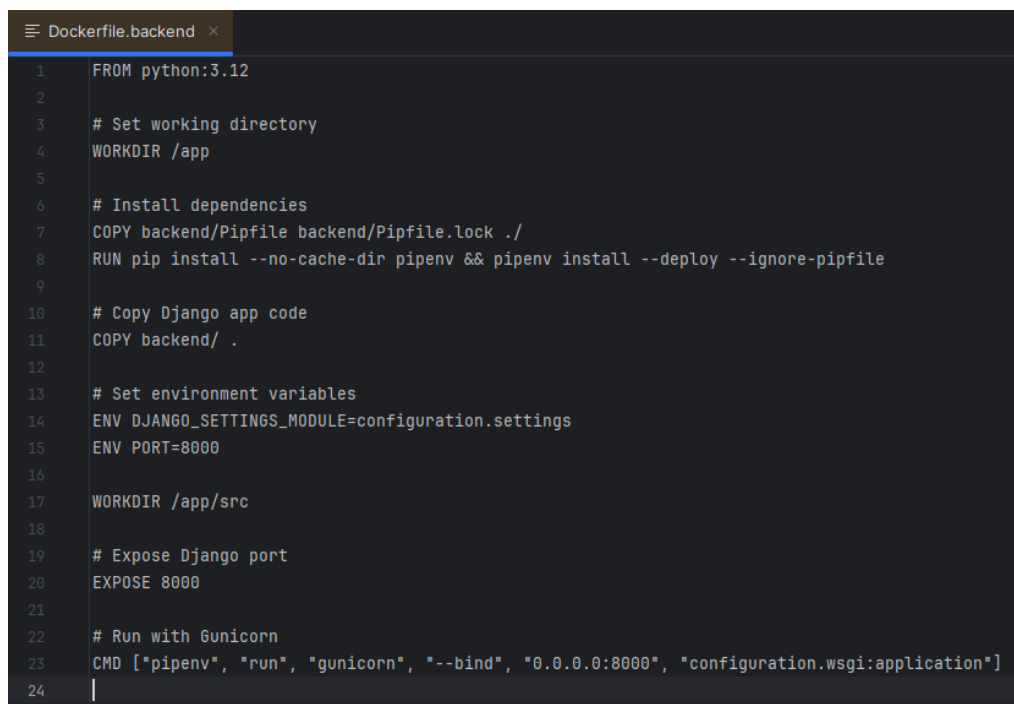
Što se tiče klijentskog dela aplikacije, njegovo pokretanje je definisano u *Dockerfile.frontend* datoteci i njegov sadržaj može da se vidi na Slici 4.1. Može da se primeti da se koristi *multi-stage build* koji služi da odvoji deo za instaliranje i kompilaciju fajlova od dela za pokretanje aplikacije. Ovo doprinosi tome da završna slika bude što manja.



```
1 FROM node:18-alpine AS builder
2
3 WORKDIR /app
4
5 COPY ../frontend/package.json ../frontend/package-lock.json ./
6
7 RUN npm install --production
8 COPY ../frontend .
9
10 RUN npm run build
11
12 # Multi-stage build
13 FROM node:18-alpine
14
15 WORKDIR /app
16
17 COPY --from=builder /app ./
18
19 EXPOSE 3000
20
21 CMD ["npm", "run", "start"]
22
```

Slika 4.1: Dockerfile za klijentski deo aplikacije

Pokretanje serverskog dela aplikacije definisano je u *Dockerfile.backend* datoteci i njegov sadržaj može da se vidi na Slici 4.2. Može da se primeti da se koristi *Gunicorn* koji predstavlja *WSGI HTTP* server za produkciona okruženja.



```
1 FROM python:3.12
2
3 # Set working directory
4 WORKDIR /app
5
6 # Install dependencies
7 COPY backend/Pipfile backend/Pipfile.lock ./
8 RUN pip install --no-cache-dir pipenv && pipenv install --deploy --ignore-pipfile
9
10 # Copy Django app code
11 COPY backend/ .
12
13 # Set environment variables
14 ENV DJANGO_SETTINGS_MODULE=configuration.settings
15 ENV PORT=8000
16
17 WORKDIR /app/src
18
19 # Expose Django port
20 EXPOSE 8000
21
22 # Run with Gunicorn
23 CMD ["pipenv", "run", "gunicorn", "--bind", "0.0.0.0:8000", "configuration.wsgi:application"]
24 |
```

Slika 4.2: Dockerfile za serverski deo aplikacije

Sadržaj glavne `docker-compose.yml` datoteke može da se vidi na Slici 4.3. Da bi se projekat pokrenuo potrebno je pozicionirati se u `/docker` direktorijum i pokrenuti sledeću komandu:

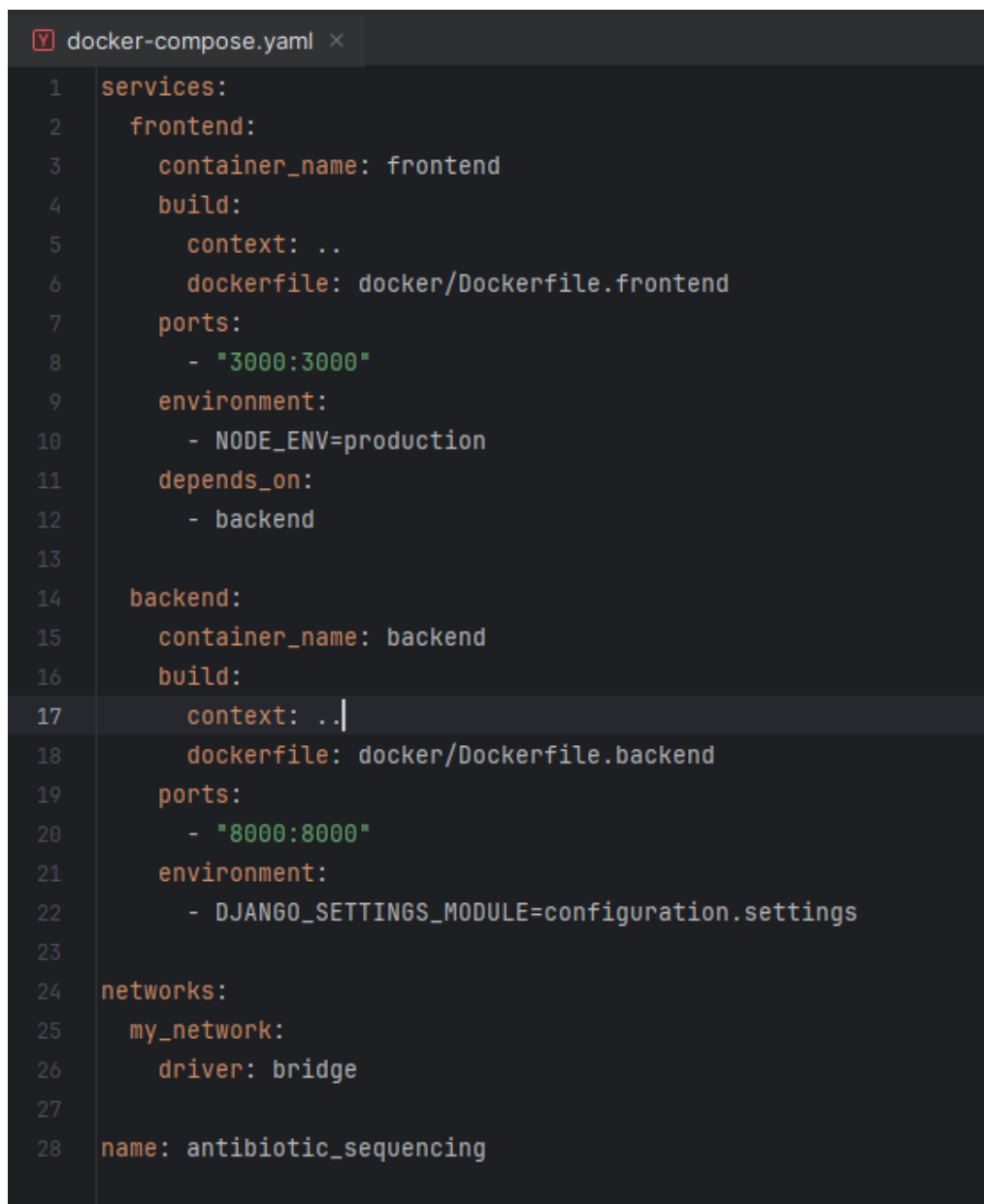
```
docker-compose up -d --build
```

Ova komanda pokreće dva kontejnera. *Frontend* delu aplikacije može da se pristupi tako što se u veb pregladaču otvori `http://localhost:3000`, dok se *backend* deo aplikacije nalazi na `http://localhost:8000`.

4.1.2 Direktno pokretanje komponenti

Da bi se direktno pokrenuo *Frontend* aplikacije potrebno je imati instaliran **Node 18** kao i **npm 10** paket menadžer. Potrebno je pozicionirati se u `/frontend` direktorijum i pokrenuti naredne komande:

```
npm install
npm run build
npm run start
```



```
1 services:
2   frontend:
3     container_name: frontend
4     build:
5       context: ..
6       dockerfile: docker/Dockerfile.frontend
7     ports:
8       - "3000:3000"
9     environment:
10      - NODE_ENV=production
11     depends_on:
12      - backend
13
14   backend:
15     container_name: backend
16     build:
17       context: ..|
18       dockerfile: docker/Dockerfile.backend
19     ports:
20       - "8000:8000"
21     environment:
22      - DJANGO_SETTINGS_MODULE=configuration.settings
23
24 networks:
25   my_network:
26     driver: bridge
27
28 name: antibiotic_sequencing
```

Slika 4.3: Docker Compose fajl koji pokreće definisane Docker fajlove

Nakon pokretanja ovih komandi klijentskom delu aplikacije može da se pristupi iz veb pregledača na adresi <http://localhost:3000>.

Da bi se direktno pokrenuo *Backend* aplikacije potrebno je imati instaliran **Python 3.12** kao i **pip** paket menadžer. Potrebno je pozicionirati se u **/backend** direktorijum i izvršiti naredne komande:

```
python -m venv venv
venv\Scripts\activate
```



```
pip install pipenv
pipenv install -d
cd src
python manage.py runserver
```

Pokretanjem ovih komandi kreiraćemo virtuelno okruženje u kom će se instalirati sve zavisnosti ove aplikacije. Za praćenje verzije korišćenih biblioteka korišćen je *Pipfile* i zato mora da se instalira i *pipenv*. Nakon pokretanja, serverskom delu aplikacije mogu se slati zahtevi na adresu *http://localhost:8000*.

4.1.3 Google Cloud Run implementacija

Aplikacija je trenutno *deploy*-ovana na Google Cloud Run i može joj se pristupiti preko *https://antibiotic-sequencing-304513663933.us-central1.run.app/*. Koristi se besplatna verzija koja se ne naplaćuje a pored toga *Google Cloud Run* pruža sledeće pogodnosti:

- **Skalabilnost:** Automatsko skaliranje u zavisnosti od opterećenja
- **Integracija:** Potpuna podrška za *Docker* kontejnere
- **Bezbednost:** Ugrađena zaštita *DDoS* napada
- **Monitoring:** Integrisani *Cloud Monitoring* alati

Postoji *Google Cloud CLI* alat koji može da se instalira i da se iz terminala pokreću odgovarajuće komande. Proces se sastoji iz sledećih koraka i potrebno je da se ovi koraci odrade i za klijentsku i za serversku komponentu:

1. Pravljenje projekta na *Google Cloud* platformi čiji će se *PROJECT_ID* dalje koristiti
2. Povezivanje *Google Cloud CLI* alata sa *Google* nalogom:

```
gcloud auth login
```

3. Definisane sa kojim projektom želimo da radimo:

```
gcloud config set project <PROJECT_ID>
```

4. Pravljenje *Docker* image-a:

```
docker build -t gcr.io/<PROJECT_ID>/fe:v1.0  
-f docker/Dockerfile.frontend .
```

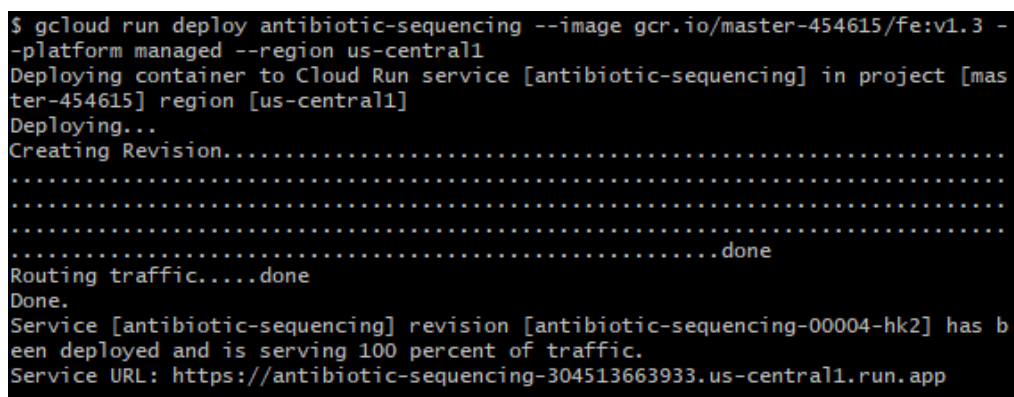
5. Kačenje slike na njihov repozitorijum:

```
docker push -t gcr.io/<PROJECT_ID>/fe:v1.0
```

6. Deploy servisa:

```
gcloud run deploy antibiotic-sequencing  
--image gcr.io/<PROJECT_ID>/fe:v1.0  
--platform managed  
--region us-central1  
--allow-unauthenticated  
--port 3000
```

Nakon toga u terminalu se dobija *URL* na kom može da se pristupi podignutom servisu što se može videti na Slici 4.4. Takođe u toj situaciji frontend komponenti mora u */.env* fajlu da se promeni *URL* koji vodi do backend komponente.



```
$ gcloud run deploy antibiotic-sequencing --image gcr.io/master-454615/fe:v1.3 -  
-platform managed --region us-central1  
Deploying container to Cloud Run service [antibiotic-sequencing] in project [mas  
ter-454615] region [us-central1]  
Deploying...  
Creating Revision.....done  
Routing traffic.....done  
Done.  
Service [antibiotic-sequencing] revision [antibiotic-sequencing-00004-hk2] has b  
een deployed and is serving 100 percent of traffic.  
Service URL: https://antibiotic-sequencing-304513663933.us-central1.run.app
```

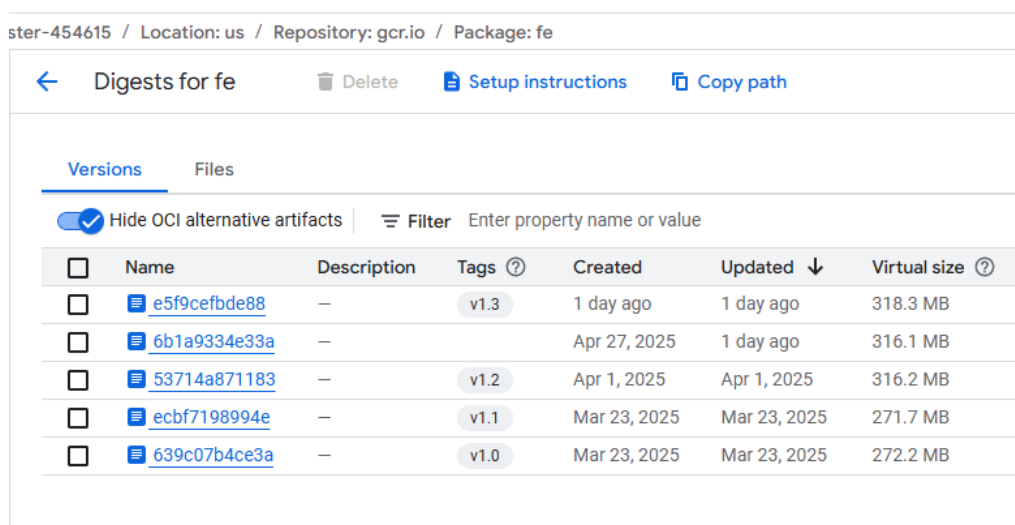
Slika 4.4: Dobijeni *URL* nakon što se aplikacija podigne na Google Cloud Run servisu

U slučaju da je potrebno okačiti novu verziju aplikacije, prate se isti koraci za pravljenje i kačenje slike dok se komanda za *deploy* razlikuje jer se navodi kako se zove komponenta koja se ažurira:

GLAVA 4. ELEKTRONSKA PLATFORMA

```
gcloud run deploy antibiotic-sequencing
--image gcr.io/<PROJECT_ID>/fe:v1.1
--platform managed
--region us-central1
```

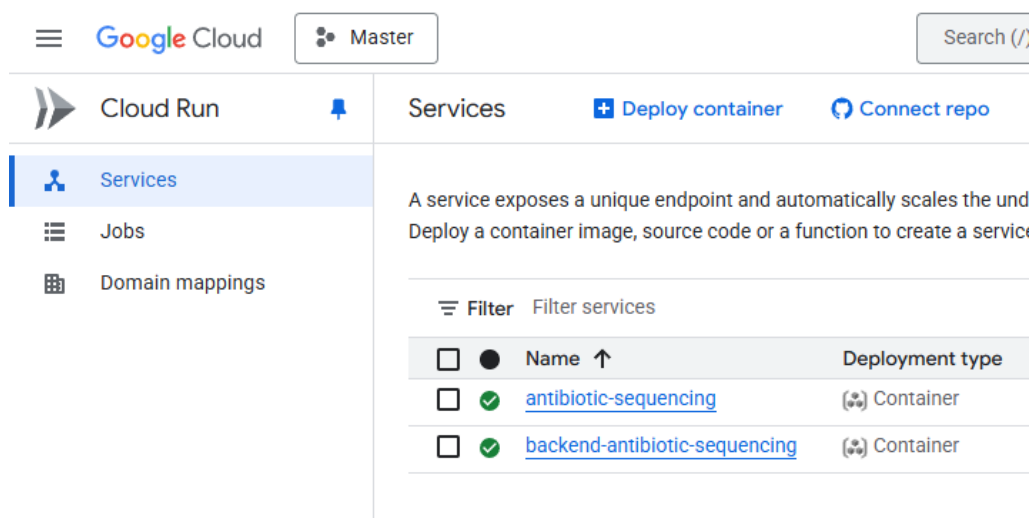
Ako želimo da pristupimo *Docker* slikama koje smo okačili potrebno je otvoriti <https://console.cloud.google.com/artifacts> (Slika 4.5).



ster-454615 / Location: us / Repository: gcr.io / Package: fe						
← Digests for fe Delete Setup instructions Copy path						
Versions Files						
<input checked="" type="checkbox"/> Hide OCI alternative artifacts Filter Enter property name or value						
<input type="checkbox"/>	Name	Description	Tags ?	Created	Updated ↓	Virtual size ?
<input type="checkbox"/>	e5f9cefbde88	—	v1.3	1 day ago	1 day ago	318.3 MB
<input type="checkbox"/>	6b1a9334e33a	—		Apr 27, 2025	1 day ago	316.1 MB
<input type="checkbox"/>	53714a871183	—	v1.2	Apr 1, 2025	Apr 1, 2025	316.2 MB
<input type="checkbox"/>	ecbf7198994e	—	v1.1	Mar 23, 2025	Mar 23, 2025	271.7 MB
<input type="checkbox"/>	639c07b4ce3a	—	v1.0	Mar 23, 2025	Mar 23, 2025	272.2 MB

Slika 4.5: Docker slike koje su okačene na Google Container Registry

Podignutim servisima može se pristupiti preko <https://console.cloud.google.com/run> (Slika 4.6).

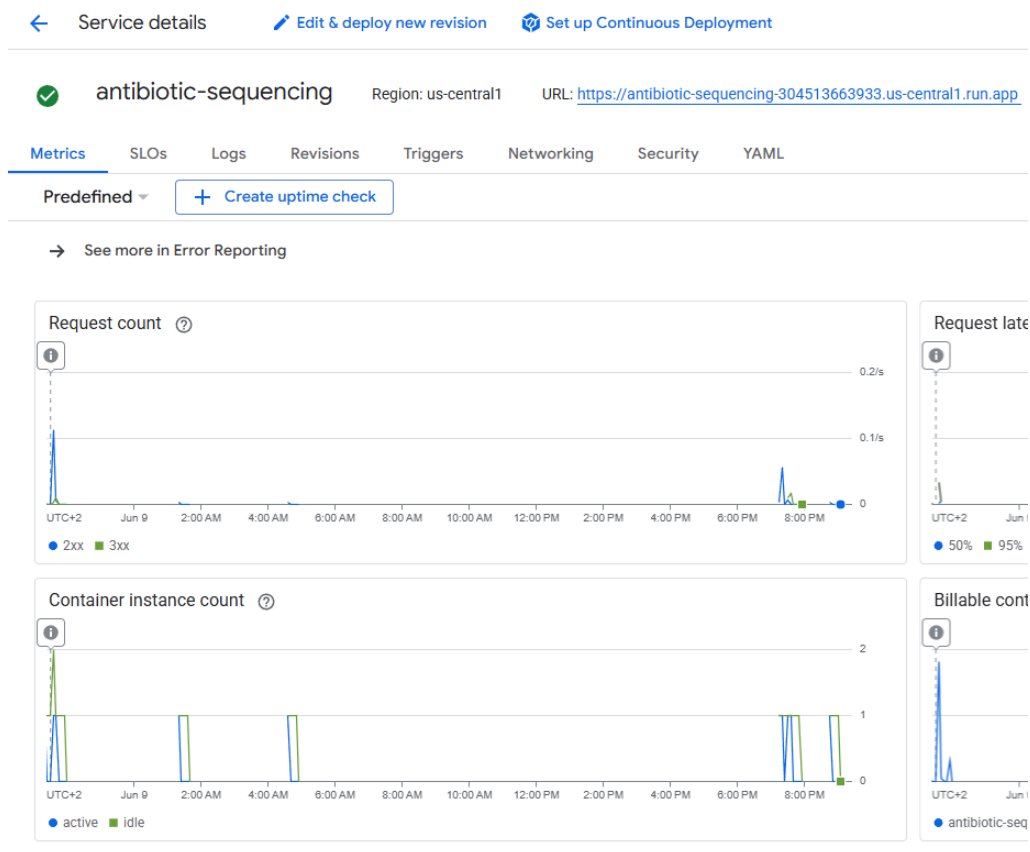


Google Cloud Master Search (/)	
Cloud Run Services Jobs Domain mappings	Services + Deploy container Connect repo
A service exposes a unique endpoint and automatically scales the und Deploy a container image, source code or a function to create a service	
Filter Filter services	
<input type="checkbox"/>	<input checked="" type="checkbox"/> Name ↑ Deployment type
<input type="checkbox"/>	<input checked="" type="checkbox"/> antibiotic-sequencing (Container)
<input type="checkbox"/>	<input checked="" type="checkbox"/> backend-antibiotic-sequencing (Container)

Slika 4.6: Pristupi servisima koji su podignuti na Google Cloud Run platformi

GLAVA 4. ELEKTRONSKA PLATFORMA

Ako se klikne na neki od servisa pristupiće se brojnim metrikama koje su nam dostupne (Slika 4.7). Takođe, tu možemo da vidimo i koji je *URL* našeg servisa kao i da vidimo koji je status i da li su se desile neke greške.



Slika 4.7: Detalji i metrike koje su nam dostupne kada pristupimo nekom servisu

4.2 Funkcionalnosti platforme

4.2.1 Početna stranica

Prilikom otvaranja aplikacije, biće prikazana početna stranica što se može videti na Slici 4.8. Na vrhu stranice nalazi se navigacioni meni kojim može da se prelazi sa stranice na stranicu. Takođe, u gornjem desnom uglu nalazi se ikonica koja vodi ka *Github* repozitorijumu gde može da se nađe izvorni kod ove aplikacije. Klikom na dugme *Istraži algoritme* odlazi se na stranicu *Uvod*, gde mogu da se nađu teorijska objašnjenja pojma koja su potrebna za razumevanje algoritama.



Slika 4.8: Početna stranica kada se otvori aplikacija

4.2.2 Uvodna stranica

Ova stranica služi da predstavi teorijske osnove i pojmove koji su potrebni za dalje razumevanje algoritama. Na Slici 4.9 može da se vidi deo ove stranice. Klikom na bilo koju sliku sa ove stranice ona će se otvoriti i omogućiti jasniji prikaz iste.

Na dnu stranice stranice, koja je prikazana na Slici 4.10, takođe postoji meni koji vodi ka algoritmima koji su obrađeni u sklopu ovde aplikacije.



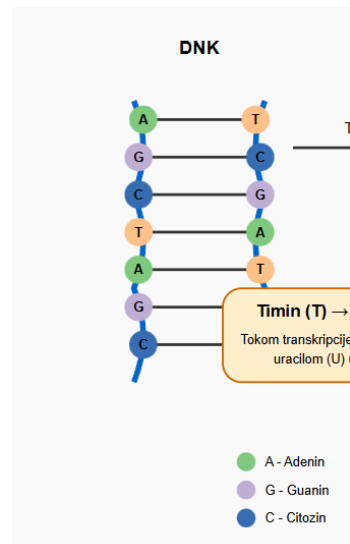
Uvod u sekvenciranje antibiotika

Proces sekvenciranja antibiotika je fundamentalan u razumevanju kako su ovi molekuli proizvedeni od strane bakterija i kako se oni mogu sintetizovati ili modifikovani za primene u medicini. Antibiotici su često peptidi - kratki proteini odnosno kratak niz aminokiselina, ali mnogo antibiotika, uglavnom neribozomalni peptidi (*non-ribosomal peptides - NRPs*), ne prati standardna pravila za sintezu proteina čime se otežava njihovo sekvenciranje [1].

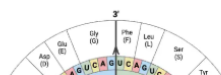
DNK sadrži recept za kreiranje proteina. Odnosno, sastoji se od gena koji mogu biti uključeni i tada će se na osnovu njih kreirati proteini ili isključeni kada se oni neće koristiti za kreiranje proteina. Isključenost ili uključenosn nekog gena zavisi od toga da li je potrebno da se kreira neki protein ili nije potrebno (npr. fotosinteza kod biljaka koja se obavlja samo preko dana).

Tradicionalno, proteini prate **Centralnu Dogmu Molekularne biologije**, koja kaže da se DNK prvo prepisuje u RNK - slika 1, a zatim se RNK prevodi u protein. Na slici jedan se može se primetiti da se DNK sastoji od 2 lanca koja su komplementarna. Enzim *RNK polimeraza* se kači na početak gena i kreće kroz gene gde razdvaja lanac i stvara prostor za prepisivanje DNK u RNK čime se dobija RNK.

Prilikom prevođenja RNK u protein potrebno je na osnovu nukleotida odrediti koja je aminokiselina u pitanju. Organela ribozom je zadužena da odradi ovaj posao i pošto je potrebno na osnovu nukleotidne sekvence uniformno odrediti koja je aminokiselina u pitanju uzima se sekvenca od 3 nukleotida takođe poznata kao kodon. Pošto je uzeta sekvenca od 3 nukleotida ovo nam daje 64 različita kodona koja treba da se prevedu u 20 aminokiselina, da smo uzeli sekvenca od 2 nukleotida dobili bismo 16 različitih kombinacija čime ne bismo mogli da dobijemo sve aminokiseline. Na slici 2 može se videti kako se kodoni prevode u odgovarajuće aminokiseline. Postoje start i stop kodoni koji određuju početak odnosno kraj sekvence koja se prevodi u protein.

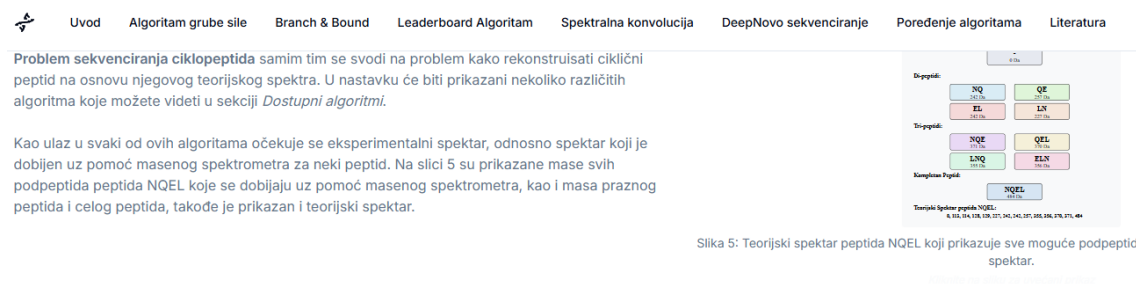


slika 1: Transkripcija DNK u RNK. Enzim RNI komplem

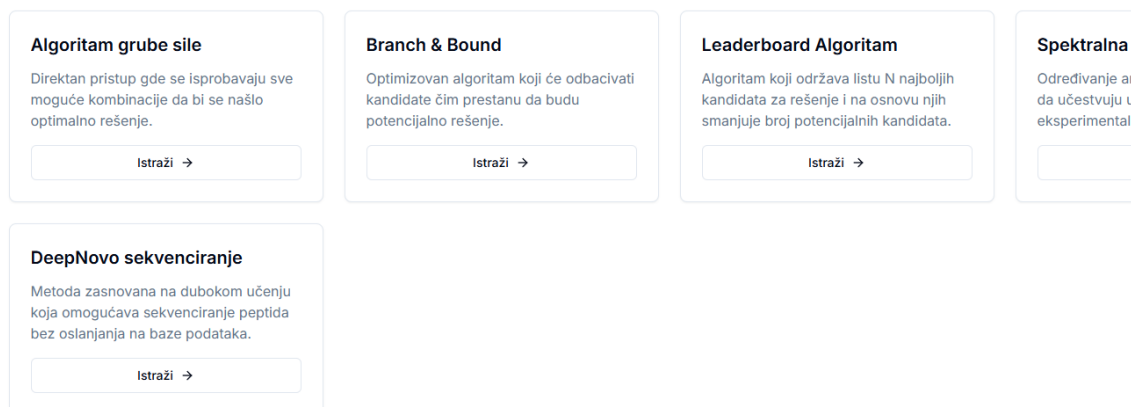


Slika 4.9: Uvodna stranica sa teorijskim pojmovima

GLAVA 4. ELEKTRONSKA PLATFORMA



Dostupni algoritmi



Slika 4.10: Navigacioni meni na dnu Uvodne stranice koji vodi ka algoritmima

4.2.3 Pristup grubom silom

Stranica za pristup grubom silom, prikazana na Slici 4.11, sadrži objašnjenje ovog algoritma kao i kod u programskom jeziku *Python* za implementaciju algoritma. Nakon objašnjenja i koda za algoritam može da se unese eksperimentalni spektar i postoji mogućnost da se odabere da li korisnik želi da vidi vizuelizaciju ovog algoritma ili želi da vidi samo rešenja 4.12. Na Slici 4.13 može da se vidi drvo koje se izgradilo da bi se došlo do rešenja. Postoje i elementi za kontrolisanje izvršavanja animacije kao što su *Play/Pause/Reset* a pored toga postoji i *slider* kojim može da se premota animacija do određenog dela izvršavanja. Pošto drvo izvršavanja može da bude veoma veliko dodata je i opcija uveličavanja tako da određeni deo drveta može da se uveliča a onda ostatak drveta može da se vidi povlačenjem miša. Čvorovi koji su obeleženi zelenom bojom predstavljaju rešenje a crveni čvorovi su odbačeni i ne predstavljaju rešenje. Kada se miš postavi iznad nekog crvenog čvora prikazaće se i objašnjenje zašto on nije rešenje. Kada se animacija završi moguće je kliknuti i na dugme *Download* čime će se izgrađeno drvo skinuti na Vaš računar u *SVG* formatu.



Pristup grubom silom

Pristup grubom silom (eng. Brute Force) [2] je najjednostavniji pristup rešavanju problema sekvenciranja antibiotika. Ovaj metod sistematski ispituje sve moguće formirati peptid zadate mase. Iako je ovaj pristup garantovano pronalazi tačno rešenje ako ono postoji, njegova vremenska složenost je eksponencijal čini nepraktičnim za duže peptide.

Na primer, za peptid mase 579 Da, algoritam će generisati sve moguće kombinacije aminokiselina i proveriti da li njihova ukupna masa odgovara zadatoj masi. Postojeći različiti peptidi sa istom ukupnom masom (FPAYT i QNWGS), što dodatno komplikuje problem.

F	147 Da	Q
P	97 Da	N
A	71 Da	W
Y	163 Da	G
T	101 Da	S
Ukupna masa:		579 Da Ukupna masa:

Da bi se utvrdilo koji od peptida je tačno rešenje, algoritam mora da generiše teorijski spektar za svaki kandidat peptid i uporedi ga sa eksperimentalnim spektrom algoritma, ali je neophodno za pronalaženje tačnog rešenja.

Naredni deo prikazuje implementaciju ovog algoritma u programskom jeziku Python.

Kod za pristup grubom silom

Algoritam na ulazu očekuje eksperimentalni spektar uređen rastuće koji uključuje 0 i masu celog peptida koji se sekvencira.

Implementacija pristupa grubom silom načinio od praznog peptida i u svakom koraku proširio peptida dodavanjem aminokiselina uz pomoć funkcije `ai`.

Slika 4.11: Stranica za pristup grubom silom sa objašnjenjem istog

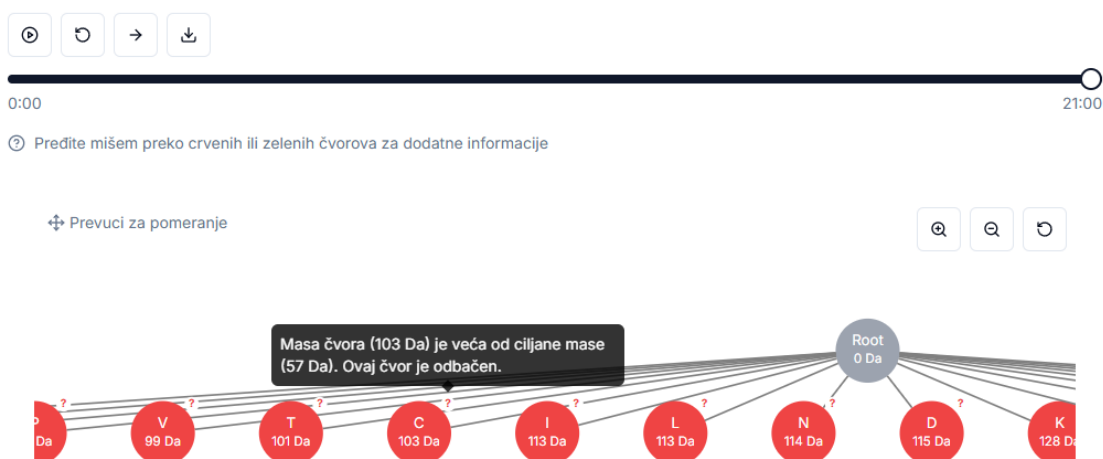
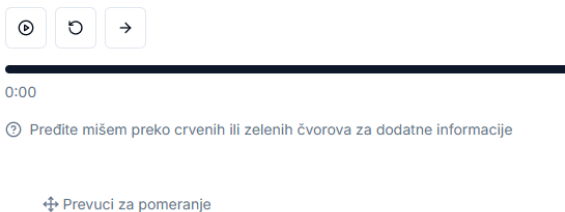
Na kraju će se za svaki peptid koji je kandidat da bude rešenje prikazati i njegov teorijski spektar, a u slučaju da se taj teorijski spektar poklapa sa unetim eksperimentalnim spektrom, prikazaće se poruka da je pronađeno rešenje, što se može videti i na Slici 4.14. U slučaju da za zadati eksperimentalni spektar nema rešenja ispisaće se poruka koja to i govori.

Primeri peptida i njihovih teorijskih spektara:

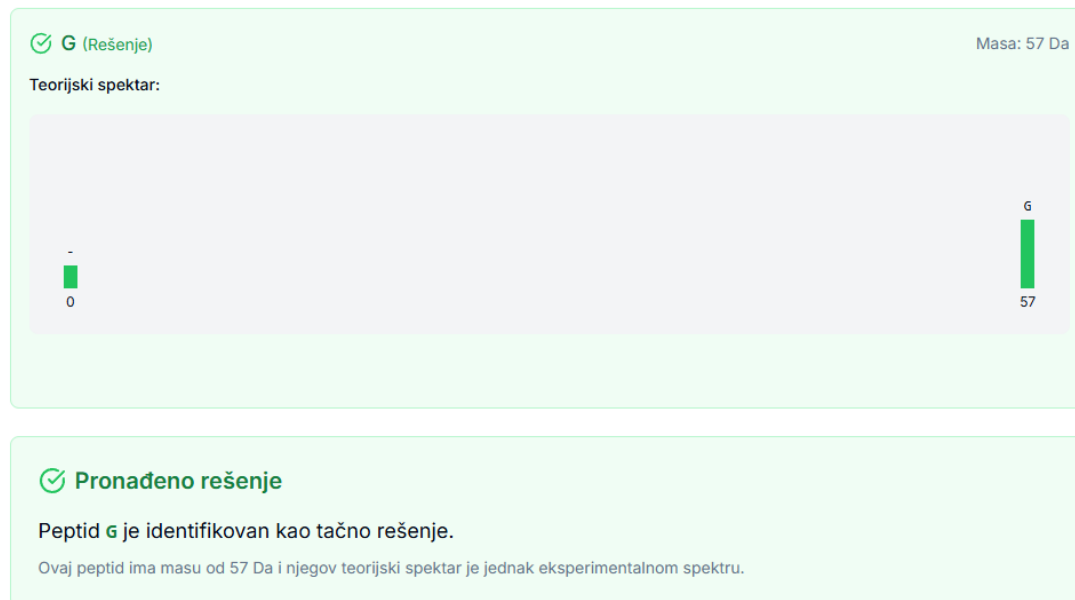
- Unesi sekvencu

☐ Prikaži samo rešenje (bez vizuelizacije)

Analiziraj sekvencu



Kandidati sa masom 57 Da:



Slika 4.14: Pronađena rešenja u pristupu grubom silom

4.2.4 Branch and Bound

Stranica za algoritam *Branch and Bound*, prikazana na Slici 4.15, sadrži objašnjenje ovog algoritma kao i kod u programskom jeziku *Python* za implementaciju algoritma.

Nakon objašnjenja i koda za algoritam može da se unese eksperimentalni spektar i može da se odabere da se vidi vizuelizacija ovog algoritma ili samo rešenje. Ovaj algoritam je veoma sličan pristupu grubom silom, tako da je i vizuelizacija i elementi koji čine tu vizuelizaciju potpuno ista. Jedina razlika je što je ovaj algoritam dosta brži i može da ranije odseče kandidate koji nisu rešenje tako da eksperimentalni spektar koji može da se unese je duži. Dodatno, pošto u ovom algoritmu postoji više razloga zašto je neki čvor odsečen tako će i poruke koje se prikazuju kada se mišem prevuče preko nekog čvora biti drugačije. Ovde je takođe po okončanju animacije moguće skinuti igrađeno drvo u *SVG* formatu.



Branch and Bound Algoritam

Branch and Bound algoritam [2] je optimizovana verzija pristupa grubom silom koja koristi strategiju "podeli pa vladaj" za efikasnije pretraživanje p silom koji ispituje sve moguće kombinacije, Branch and Bound algoritam inteligentno eliminiše delove prostora pretrage koji ne mogu sadržati optin

U kontekstu sekvenciranja peptida, algoritam funkcioniše na sledeći način:

- **Grananje (Branch):** Algoritam gradi stablo pretrage gde svaki čvor predstavlja delimičnu sekvencu peptida. Svaki čvor se grana dodavanjem no
- **Ograničavanje (Bound):** Za svaki čvor, algoritam procenjuje da li taj put može dovesti do validnog rešenja. Ako masa peptida već premašuje cilj sekvence peptida nije konzistentan sa eksperimentalnim, ta grana se "odseca" i dalje ne istražuje.
- **Optimizacija:** Algoritam može koristiti dodatne heuristike za procenu koje grane prvo istražiti, što dodatno ubrzava pronalaženje rešenja.

Prednosti Branch and Bound algoritma u odnosu na pristup grubom silom su značajne:

Pristup grubom silom

- Istražuje sve moguće kombinacije
- Eksponencijalna vremenska složenost
- Garantuje pronalaženje svih rešenja
- Neefikasan za duže peptide

Branch and Bound

- Inteligentno eliminiše neperspektivne gran
- Značajno bolja vremenska složenost (u na složenosti ali i dalje dosta brži)
- I dalje garantuje pronalaženje svih rešenja
- Efikasniji za duže peptide

Kod za Branch and Bound algoritam

Algoritam na ulazu očekuje eksperimentalni spektar uređen rastuće koji uključuje 0 i masu celog peptida koji se sekvencira. Za razliku od pristu koristi dodatne provere da bi eliminisao neperspektivne grane što ranije.

```
def branch_and_bound(target_spectrum):  
    peptides = ['']  
    results = []
```

Slika 4.15: Stranica za algoritam Branch and Bound sa objašnjenjem istog

4.2.5 Leaderboard

Stranica za algoritam *Leaderboard*, prikazana na Slici 4.16, sadrži objašnjenje ovog algoritma kao i kod u programskom jeziku *Python* za implementaciju algoritma.



Leaderboard Algoritam

Leaderboard algoritam [2] je optimizovani pristup za sekvenciranje peptida. Za razliku od sekvenciranja grubom silom koje zahteva tačno poklapanje između eksperimentalnog spektra, ovaj algoritam je dizajniran da radi sa **nedostajućim** i **lažnim masama** tako što prati samo najbolje kandidate peptida umesto svih

U realnim eksperimentalnim podacima masene spektrometrije, izmereni eksperimentalni spektar je često zašumljen i nepotpun. Neki očekivani fragmenti mogu pojaviti zbog pozadinskog šuma. Savršeno poklapanje između teorijskog spektra peptida i eksperimentalnog spektra je malo verovatno.

Prednosti Algoritma



Efikasnost

Fokusira se samo na najperspektivnije kandidate, značajno smanjujući vreme izvršavanja



Skalabilnost

Efikasno radi sa peptidima različitih dužina bez eksponencijalnog rasta vremena



Preciznost

Održava visoku tačnost uprkos šumu u eksperimentalnim podacima

Primene



Eksperimentalni Podaci

Idealan za analizu realnih podataka masene spektrometrije



Nepotpuni Podaci

Kada tačno poklapanje nije moguće zbog nepotpunosti podataka



Vremenski Kritične Analize

Kada je potrebna brza identifikacija peptida iz velikih skupova podataka

Kod za Leaderboard algoritam

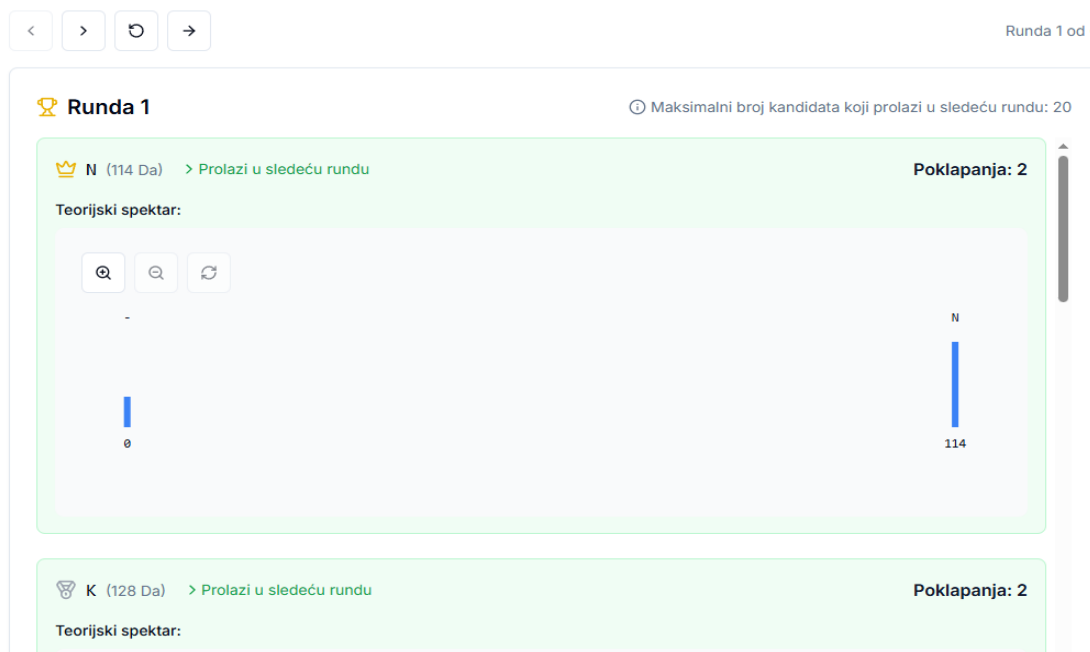
Algoritam održava listu najboljih kandidata (leaderboard) i u svakoj iteraciji proširuje samo najperspektivnije peptide. Ovo značajno smanjuje prostor pretrage. Za svaki od peptida koji se generiše određujemo koji je njegov *score*, odnosno broj masa linearnog spektra peptida koji je jednak masama u eksperimentalnom spektru.

```
def leaderboard_sequencing(target_spectrum):
    # Krećemo od praznog peptida
    peptides = ['']

    # Peptid koji je trenutno najbolji kandidat i njegov score
    leader_peptide = ''
```

Slika 4.16: Stranica za algoritam Leaderboard sa objašnjenjem istog

Nakon objašnjenja i koda za algoritam može da se unese eksperimentalni spektar i može da se odabere da se vidi vizuelizacija ovog algoritma ili da se prikažu samo rešenja. Pošto ovaj algoritam funkcioniše po principu prolaska u sledeće runde vizuelizacija za ovaj algoritam je tako i odrađena što se može videti na Slici 4.17. Za kontrolu animacije postoje strelice za prebacivanje iz runde u rundu, strelica da nas prebaci skroz u poslednju rundu da bismo videli koje je rešenje kao i dugme za resetovanje odnosno prebacivanje na prvu rundu. Uz svaki peptid piše njegova masa, prikazuje se njegov teorijski spektar kao i broj poklapanja teorijskog spektra sa zadatim eksperimentalnim spektrom. Svaki peptid koji prolazi u sledeću rundu je označen zelenom bojom uz tekst da je prošao dalje. Teorijski spektar svakog od peptida može da se uveliča po potrebi i da se koristi *drag and drop* mehanizam da se vidi neki određeni deo spektra. Da bi se poboljšale performanse vizuelizacije za prikaz svih mogućih kandidata u nekoj rundi korišćen je princip *lazy loading*, odnosno tek kada se lista spušta na dole prikazaće se sledeći kandidati koji su razmatrani



Slika 4.17: Prikaz rundi za Leaderboard algoritam

u toj rundi.

U poslednjoj rundi biće prikazani peptidi koji su rešenje zadatog eksperimentalnog spektra i to je prikazano na Slici 4.18. Za peptide koji nisu rešenje u nekoj rundi prevlačenjem miša preko njih biće prikazana i poruka o razlogu zašto nisu rešenje.

<
>
↺
→

Runda 6 od 6

🏆 Runda 6

✓ **Ukupno različitih rešenja: 52**

NKE Masa: 371 Da Poklapanja: 8	NQE Masa: 371 Da Poklapanja: 8	NEK Masa: 371 Da Poklapanja: 8
NEQ Masa: 371 Da Poklapanja: 8	KNE Masa: 371 Da Poklapanja: 8	KEN Masa: 371 Da Poklapanja: 8
QNE Masa: 371 Da Poklapanja: 8	QEN Masa: 371 Da Poklapanja: 8	ENK Masa: 371 Da Poklapanja: 8
ENQ Masa: 371 Da Poklapanja: 8	EKN Masa: 371 Da Poklapanja: 8	EQN Masa: 371 Da Poklapanja: 8
NEGA Masa: 371 Da Poklapanja: 8	NEAG Masa: 371 Da Poklapanja: 8	KEGG Masa: 371 Da Poklapanja: 8
QEGG Masa: 371 Da Poklapanja: 8	ENG Masa: 371 Da Poklapanja: 8	ENAG Masa: 371 Da Poklapanja: 8

Slika 4.18: Poslednja runda i peptidi koji su rešenje za Leaderboard algoritam

4.2.6 Spektralna konvolucija

Stranica za algoritam spektralne konvolucije, prikazana na Slici 4.19, sadrži objašnjenje ovog algoritma kao i kod u programskom jeziku *Python* za implementaciju algoritma.



Spektralna konvolucija

Spektralna konvolucija [2] je tehnika koja se koristi za identifikaciju aminokiselina koje mogu biti prisutne u peptidu na osnovu eksperimentalnog spektra. Ovi spektru i identifikuje one koje odgovaraju masama aminokiselina.

Proces se sastoji iz dva glavna koraka:

1. **Izračunavanje konvolucije:** Za svaki par masa u spektru, izračunava se njihova razlika. Ove razlike mogu odgovarati masama aminokiselina.
2. **Identifikacija aminokiselina:** Najčešće razlike koje se pojavljuju u spektru verovatno odgovaraju aminokiselinama prisutnim u peptidu. Te aminokiseline se algoritmu.

Glavna prednost ovog algoritma jeste to što u samom startu smanjuje skup aminokiselina koje mogu da učestvuju u građenju peptida, čime se algoritam dostiže mogućnost da se identifikuju nepoznate ili modifikovane aminokiseline. Još jedna od prednosti ovog algoritma jeste to što može da radi na eksperimentalnim nedostajućih masa.

Kod za algoritam spektralne konvolucije

Algoritam spektralne konvolucije računa razlike između svih parova masa u eksperimentalnom spektru, a zatim identifikuje najčešće razlike koje odgovaraju aminokiselinama. Zatim se koriste za generisanje kandidata peptida.

```
def spectral_convolution(target_spectrum):
    num_of_el_in_spectrum = len(target_spectrum)
    convolution = []

    # Elemente spektra možemo da posmatramo kao matricu, gde prvu vrstu i prvu kolonu predstavljaju elementri eksperimentalnog spektra.
    # Prolazimo kroz elemente spektra i međusobno ih poredimo, kao kroz donju trougaonu matricu i ako je razlika u okviru datog opsega onda je
    # jedna aminokiselina koja može a učestvuje u rešenju
    for i in range(num_of_el_in_spectrum):
        for j in range(i):
            diff = target_spectrum[i] - target_spectrum[j]
            if 57 <= diff <= 200:
                convolution.append(diff)

    # Određujemo broj pojavljivanja masa
```

Slika 4.19: Stranica za algoritam spektralne konvolucije sa objašnjenjem istog

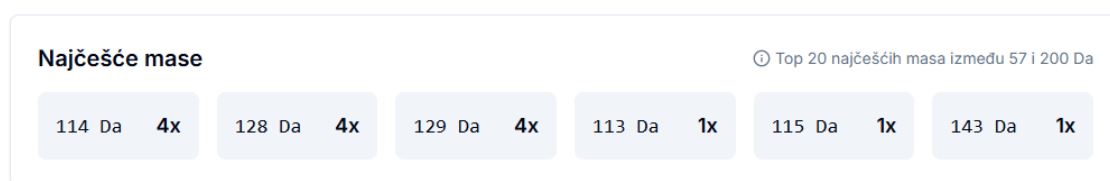
Nakon objašnjenja i koda za algoritam može da se unese eksperimentalni spektar i može da se odabere da se vidi vizuelizacija ovog algoritma ili da se prikažu samo rešenja. Ovaj algoritam je veoma sličan *Leaderboard* algoritmu, tako da je i vizuelizacija i elementi koji čine tu vizuelizaciju potpuno ista. Jedina razlika je što ovaj algoritam na samom početku kreira matricu konvolucije (Slika 4.20), i uz pomoć matrice smanjuje broj aminokiselina koje mogu da učestvuju u izgradnji peptida (Slika 4.21). Postoje posebne kontrole za upravljanje animacijom za izgradnju matrice konvolucije, kao i posebne kontrole za *Leaderboard* deo ovog algoritma. Ove kontrole su slične onima iz prethodnih algoritama.

Matrica konvolucije



Slika 4.20: Matrica konvolucije u algoritmu spektralne konvolucije

Aminokiseline u peptidima



Slika 4.21: Najčešće mase aminokiselina koje se pojavljuju u matrici konvolucije

4.2.7 DeepNovo

Stranica za *DeepNovo* sekvenciranje, prikazana na Slici 4.22, sadrži *tab*-ove koji pružaju uvid u detalje kao što su:

- **Postojeće tehnike i problemi** - koje su postojeće tehnike i problemi sekvenciranja

- **DeepNovo** - šta je *DeepNovo* i kako funkcioniše
- **Arhitektura** - koje komponente čine *DeepNovo*
- **Rezultati** - uspešnost primene *DeepNovo* metode



Metoda za De Novo Sekvenciranje Peptida Zasnovana na Dubokom Učenju

Pristup sekvenciranju peptida koji koristi duboko učenje za identifikaciju sekvenci aminokiselina bez oslanjanja na baze podataka

Postojeće tehnike i problemi

DeepNovo

Arhitektura

Rezultati

Postojeće tehnike

Trenutne tehnike za sekvenciranje peptida (poput pretraživanja baze podataka, *de novo* sekvenciranja, kao i raznih algoritamskih pristupa) mogu imati poteškoća u radu sa novim, složenim ili nepotpunim podacima. Pristup pretraživanja baze podataka oslanja se na poređenje eksperimentalnih podataka sa bazom podataka poznatih proteinskih sekvenci, ali neki od problema u ovom pristupu su sledeći:



Nepoznati proteini

Novi proteini koji nikada ranije nisu viđeni neće se podudarati ni sa čim u bazi podataka, što dovodi do nemogućnosti identifikacije.



Nedostajući podaci

Podaci generisani iz eksperimenata masene spektrometrije mogu biti pogrešni i nepotpuni, što otežava pouzdano poređenje.

Pristup *de novo* sekvenciranja pokušava izgraditi sekvencu peptida iz početka, bez oslanjanja na bazu podataka i koristeći samo podatke koji su dobijeni masenom spektrometrijom. U okviru ovog pristupa koriste se i različiti algoritmi, kao što su pristup grubom silom, *Branch and Bound*, *Leaderboard* algoritam, kao i spektralna konvolucija, koji pokušavaju da generišu sekvence peptida i da

Slika 4.22: Stranica za *DeepNovo* sekvenciranje

4.2.8 Poređenje algoritama

Stranica za poređenje algoritama prikazuje vremena izvršavanja pristupa grubom silom, algoritama *Branch and Bound*, *Leaderboard* i spektralna konvolucija. Dodatno, za svaki algoritam prikazuje se i kolona *Status* koja govori da li je algoritam uspeo da pronađe rešenje. U slučaju da je algoritam pronašao rešenja, prikazaće se i koliko ih je pronašao. Ukoliko postoji više od 4 rešenja za određeni algoritam, biće prikazana prva 4 rešenja kao i dugme koje kada se klikne vodi na sva rešenja

za traženi algoritam. Na kraju će za svaki od algoritama biti prikazana i njegova rešenja.

U slučaju da se za teorijski spektar unese [0, 114, 128, 129, 242, 243, 257, 371] svi algoritmi pronalaze rešenja što je i prikazano na Slici 4.23.

Poređenje vremena izvršavanja

Algoritam	Vreme izvršavanja (u sekundama)	Broj pronađenih rešenja	Pregled rešenja	Status
Pristup grubom silom	0.1469	12	<div>NKE NQE NEK</div> <div>NEQ</div> <div>▼ Prikaži sve (12)</div>	✔ Uspešno
Branch and Bound algoritam	0.0006	12	<div>NKE NQE NEK</div> <div>NEQ</div> <div>▼ Prikaži sve (12)</div>	✔ Uspešno
Leaderboard algoritam	0.0165	52	<div>QNE QEN NQE</div> <div>NKE</div> <div>▼ Prikaži sve (52)</div>	✔ Uspešno
Algoritam spektralne konvolucije	0.0015	12	<div>QNE QEN NQE</div> <div>NKE</div> <div>▼ Prikaži sve (12)</div>	✔ Uspešno

Analiza performansi

Najbrži algoritam: Branch and Bound (0.0006)

Ukupno različitih rešenja: 52

Slika 4.23: Prikaz poređenje brzine algoritama kada svi algoritmi pronalaze rešenja

Kada se unese teorijski spektar koji ima lažne ili nedostajuće mase, na primer [0, 113, 114, 128, 129, 227, 240, 242, 257, 355, 356, 370, 370, 484], samo

Leaderboard algoritam i algoritam spektralne konvolucije pronalaze rešenja (Slika 4.24).

Poređenje vremena izvršavanja

Algoritam	Vreme izvršavanja (u sekundama)	Broj pronađenih rešenja	Pregled rešenja	Status
Pristup grubom silom	4.3250	0	Nema rešenja	Bez rešenja
Branch and Bound algoritam	0.0023	0	Nema rešenja	Bez rešenja
Leaderboard algoritam	0.0060	24	<div>QNLE QNIE</div> <div>QELN QEIN</div> <div>▼ Prikaži sve (24)</div>	Uspešno
Algoritam spektralne konvolucije	0.0022	24	<div>QNLE QNIE</div> <div>QELN QEIN</div> <div>▼ Prikaži sve (24)</div>	Uspešno

Analiza performansi

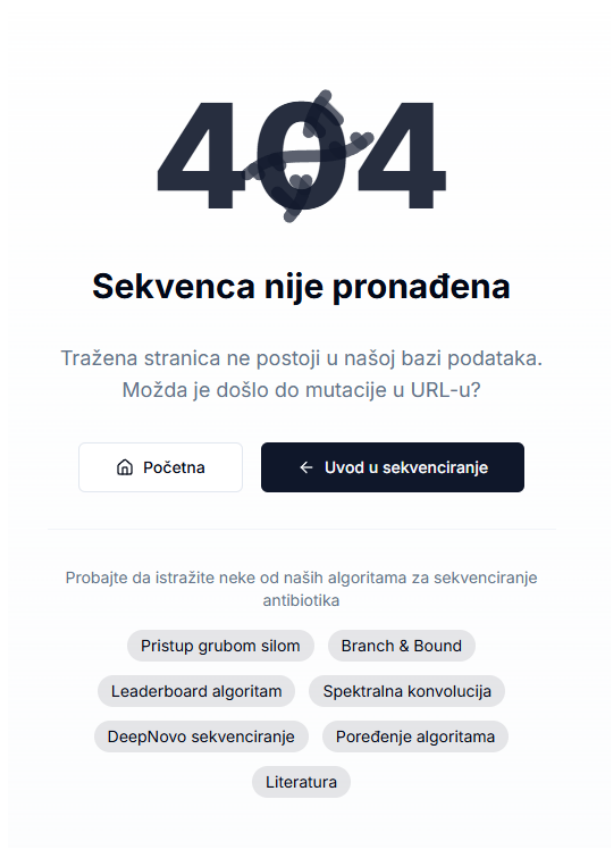
Najbrži algoritam:
Spektralna konvolucija (0.0022)

Ukupno različitih rešenja:
24

Slika 4.24: Prikaz poređenje brzine algoritama kada samo Leaderboard algoritam i algoritam spektralne konvolucije pronalaze rešenja

4.2.9 Nepostojeća stranica

U slučaju da se unese pogrešan URL za stranicu prikazaće se da ta stranica ne postoji i biće ponuđen izbor postojećih stranica, što je i prikazano na Slici 4.25.



Slika 4.25: Stranica koja se prikazuje u slučaju da je pogrešan URL unet

Glava 5

Zaključak

Razvijena elektronska platforma za sekvenciranje antibiotika predstavlja edukativni alat koji objedinjuje teorijske koncepte sa praktičnom implementacijom. Kroz interaktivne simulacije i vizuelizacije, platforma omogućava dubinsko razumevanje kompleksnih algoritama sekvenciranja.

Glavni doprinosi ovog rada uključuju:

- Integraciju više algoritama za sekvenciranje u jedinstvenu platformu
- Vizuelizaciju koraka algoritama u realnom vremenu
- Kreiranje interaktivnog okruženja za učenje
- Poboljšanje dostupnosti bioinformatičkih alata studentima

Budući radovi mogu se usredsrediti na proširenje platforme sa dodatnim algoritmima, poboljšanje performansi postojećih implementacija i integraciju sa stvarnim eksperimentalnim podacima iz masene spektrometrije.

Bibliografija

- [1] Django dokumentacija, 2025. on-line at: <https://docs.djangoproject.com/en/5.2/>.
- [2] Docker Compose dokumentacija, 2025. on-line at: <https://docs.docker.com/compose/>.
- [3] Docker dokumentacija, 2025. on-line at: <https://docs.docker.com/manuals/>.
- [4] GitHub repozitorijum sa izvornim kodom aplikacije, 2025. on-line at: <https://github.com/Milak9/master>.
- [5] Google Cloud Run dokumentacija, 2025. on-line at: <https://cloud.google.com/run/docs/overview/what-is-cloud-run>.
- [6] NextJS dokumentacija, 2025. on-line at: <https://nextjs.org/docs>.
- [7] Node 18 dokumentacija, 2025. on-line at: https://devdocs.io/node~18_lts/.
- [8] Python 3.12 dokumentacija, 2025. on-line at: <https://docs.python.org/3.12/index.html>.
- [9] TypeScript dokumentacija, 2025. on-line at: <https://www.typescriptlang.org/docs/>.
- [10] Pavel Pevzner Ari Frank. PepNovo: De Novo Peptide Sequencing via Probabilistic Network Modeling. *Analytical Chemistry*, 77:964–973, 2005.
- [11] Muhammad Zubair Eshita Garg. Mass Spectrometer, 2024. on-line at: <https://www.ncbi.nlm.nih.gov/books/NBK589702/>.

- [12] Jovana Kovačević. Materijal sa predavanja „Uvod u bioinformatiku”, 2022. on-line at: https://www.bioinformatika.matf.bg.ac.rs/predavanja/Chapter_4.pdf.
- [13] Bin Ma. Novor: Real-Time Peptide de Novo Sequencing Software. *Journal of The American Society for Mass Spectrometry*, 26:1885–1894, 2015.
- [14] Pavel Pevzner Philip Compeau. Bioinformatics Algorithms: An Active Learning Approach Vol. I, Chapter 4: How Do We Sequence Antibiotics?, 2015. on-line at: <https://cogniterra.org/course/64/promo>.
- [15] Sanju Tamang. Codon Chart: Table, Amino Acids & RNA Wheel Explained, 2024. on-line at: <https://microbenotes.com/codon-chart-table-amino-acids/>.
- [16] Jing et al. Zhang. PEAKS DB: De Novo Sequencing Assisted Database Search for Sensitive and Accurate Peptide Identification. *Molecular & Cellular Proteomics*, 11, 2011.
- [17] Xin et al. Zhang. DeepNovo: De novo peptide sequencing by deep learning. *PNAS*, 114:8247–8252, 2017.

Biografija autora

Miloš Milaković rođen je 6. avgusta 1998. godine u Beogradu. Smer Informatika na Matematičkom fakultetu Univerziteta u Beogradu upisao je 2017. godine, a završio 2021. godine sa prosečnom ocenom 8.54. Nakon toga je upisao master studije na istom smeru. Od septembra 2021. do marta 2024. godine je zaposlen na poziciji *Software developer* u firmi **Endava**. Od marta 2024. godine do sada je zaposlen na poziciji *Software developer* u firmi **LotusFlare**. Projekti na kojima je radio su uglavnom bili zasnovani na veb tehnologijama (*Telco* industrija i *FinTech* industrija), a osnovni programski jezik u kojem su projekti rađeni su *Python*, *Lua* i *TypeScript*.